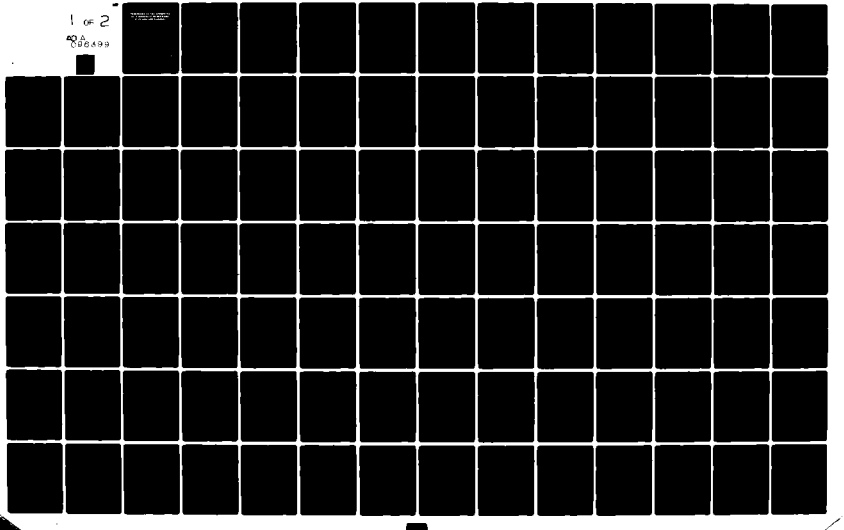
AD-A096 399 PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING F/G 9/2
PROPERTIES OF THE AUGMENTED DATA MANIPULATOR NETWORK IN A SIMD --ETC(U)
DEC 80 G B ADAMS, H J SIEGEL AFOSR-78-3581
UNCLASSIFIED TR-EE 80-51 AFOSR-TR-81-0203 NL

1 OF 2
AD A096399



PROPERTIES OF THE AUGMENTED DATA MANIPULATOR NETWORK IN AN SIMD ENVIRONMENT

George B. Adams III
Howard Jay Siegel

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 80-51
December 1980

This work was supported by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant number AFOSR-78-3581.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: 19 18 AFOSR-TR-81-0203	2. GOVT ACCESSION NO. AD-A096 399	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Properties of the Augmented Data Manipulator Network in an SIMD Environment	5. TYPE OF REPORT & PERIOD COVERED 9 Interim Sept 1,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) George B. Adams, III / Howard Jay Siegel 15	8. CONTRACT OR GRANT NUMBER(s) AFOSR-78-3581
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Electrical Engineering (12) 107 Purdue University West Lafayette, IN 47907	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) 17 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS United States Air Force, Air Force Office of Scientific Research, Building 410, Bolling AFB, Washington, DC 20332 (11)	12. REPORT DATE Dec 1989
	13. NUMBER OF PAGES 105
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (14) TR-EE 81-51	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
augmented data manipulator, computer architecture, dynamically reconfigurable systems, interconnection network, parallel processing, PASM, permutation network, SIMD machine

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Please see reverse side.

DD FORM 1 JAN 73 1473 292 000

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The demand for computers with ever greater throughput coupled with the decreased costs accompanying advances in semiconductor technology has created a great deal of interest in parallel processing systems. Single instruction stream - multiple data stream (SIMD) machines and multiple instruction stream - multiple data stream (MIMD) machines are two types of parallel processing system architectures. PASM is a partitionable SIMD/MIMD parallel processor, intended to operate in either mode of parallelism, being developed at Purdue University. The interconnection network chosen for this system will greatly influence its performance. The Generalized Cube and the Augmented Data Manipulator (ADM) are two networks being considered for use in PASM. This work is primarily concerned with the capabilities of the ADM network in SIMD mode.

The number of data permutations passable by the ADM network is explored. First the number of permutations performable by any stage is counted. Using partitioning properties of the network and combinatorial mathematics, this result is extended to permutations performable by the entire network. For N = 8 an exact count of the number of performable permutations is given. For N > 8, upper and lower bounds are given. Comparison with the Generalized Cube network is made.

Routing tag schemes are described for both the Generalized Cube and ADM networks. The number of data permutations passable by the ADM network using positive dominant or negative dominant permutation routing tags is counted. The number of permutations passable using natural permutation routing tags is bounded.

Algorithms for determining permutation passability in the ADM network using three related types of routing tags for distributed network control are presented. Correctness proofs are given and algorithm complexity determined.

To further investigate ADM network capabilities in SIMD mode, group theory is used to derive additional properties. It is shown that the ADM network cannot pass all even permutations when $N \geq 8$.

PROPERTIES OF THE AUGMENTED DATA MANIPULATOR NETWORK IN AN SIMD ENVIRONMENT

George B. Adams III

Howard Jay Siegel

School of Electrical Engineering

Purdue University

West Lafayette, Indiana 47907

Purdue University

TR-EE 80-51

December, 1980

This work was supported by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant number AFOSR-78-3581.

ACKNOWLEDGEMENTS

The authors wish to thank the following people for their comments on the contents of this work: Professor Leah J. Siegel, Professor Philip H. Swain, Professor Carl H. Smith, and Robert J. McMillen. The authors also thank Mellanie Boes for typing the manuscript. This was supported by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant number AFOSR-78-3581.

This report is based on the Master's Thesis of George B. Adams III. Portions of this work have appeared in the following:

S. D. Smith, H. J. Siegel, R. J. McMillen, and G. B. Adams III, "Use of the augmented data manipulator multistage network for SIMD machines," 1980 International Conference on Parallel Processing, Aug. 1980, pp. 75-78.

R. J. McMillen, G. B. Adams III, and H. J. Siegel, "Permuting with the augmented data manipulator network," Eighteenth Annual Allerton Conference on Communication, Control, and Computing, Oct. 1980, to appear in the proceedings.

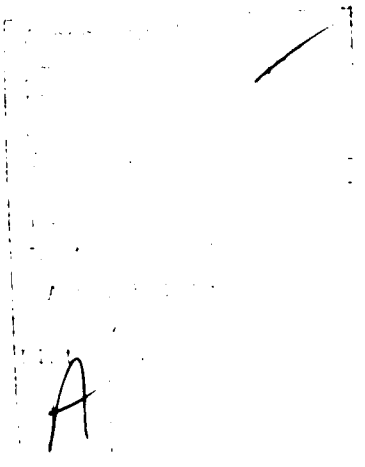


TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

The demand for computers with ever greater throughput coupled with the decreased costs accompanying advances in semiconductor technology has created a great deal of interest in parallel processing systems. Single instruction stream - multiple data stream (SIMD) machines and multiple instruction stream - multiple data stream (MIMD) machines are two types of parallel processing system architectures. PASM is a partitionable SIMD/MIMD parallel processor, intended to operate in either mode of parallelism, being developed at Purdue University. The interconnection network chosen for this system will greatly influence its performance. The Generalized Cube and the Augmented Data Manipulator (ADM) are two networks being considered for use in PASM. This work is primarily concerned with the capabilities of the ADM network in SIMD mode.

The number of data permutations passable by the ADM network is explored. First the number of permutations performable by any stage is counted. Using partitioning properties of the network and combinatorial mathematics, this result is extended to permutations performable by the entire network. For N = 8 an exact count of the number of performable permutations is given. For N > 8, upper and lower bounds are given. Comparison with the Generalized Cube network is made.

Routing tag schemes are described for both the Generalized Cube and ADM networks. The number of data permutations passable by the ADM net-

work using positive dominant or negative dominant permutation routing tags is counted. The number of permutations passable using natural permutation routing tags is bounded.

Algorithms for determining permutation passability in the ADM network using three related types of routing tags for distributed network control are presented. Correctness proofs are given and algorithm complexity determined.

To further investigate ADM network capabilities in SIMD mode, group theory is used to derive additional properties. It is shown that the ADM network cannot pass all even permutations when $N \geq 8$.

CHAPTER 1

INTRODUCTION

Throughput has been, and remains, a major limiting factor of the scope of data processing tasks performed by computer systems. Many tasks of current interest such as machine vision, image processing, seismic exploration, air traffic control, and aerodynamic simulation could greatly benefit from performance that is in excess of current computer systems.

Historically, computer system designers have attempted to meet the demand for increased throughput by building new generations of machines which, most often, differed from their predecessors only in circuit switching speed. System architecture remained reasonably similar to the basic von Neumann machine. To continue the significant gains made over the years using this approach will require further major reductions in circuit switching times. Indeed, circuits using the Josephson effect promise to make picosecond switching times practical in the not too distant future. But, there is an ultimate limit to the switching speed of a given circuit, determined by the propagation speed of electromagnetic waves: the speed of light. So alternate methods of improving throughput are of interest.

Throughput is not directly dependent on the circuit switching speed. Ultimately, throughput is maximized on a given system when any task takes only one instruction cycle to execute. To achieve a

reduction in the number of system instruction cycles, new machine organizations and algorithm structures may be used.

In tandem with the improvements in circuit switching speeds, significant circuit cost reductions have been realized. The reduced cost of hardware has made large scale parallel processing systems feasible. Such architectures are suited for problems that can be decomposed into independent subtasks. Simultaneous execution of these subtasks allows a reduction in the number of system instruction cycles needed to perform the task. All of the problems listed previously as being computationally intensive could benefit from parallel processing systems.

One type of parallel architecture is the single instruction stream - multiple data stream (SIMD) system. Such machines typically consist of N processors, N memories, an interconnection network, and a control unit. The control unit broadcasts instructions to all processing elements, and all active processors execute the same instruction simultaneously. This is the single instruction stream. Each processor executes these instructions on data stored in a memory with which it is associated. This provides the multiple data stream. The interconnection network serves to provide interprocessor communication.

A second type of parallel processor system is the multiple instruction stream - multiple data stream (MIMD) machine. Again there are typically N processors, N memories, and an interconnection network. However, processors execute instructions from their own memories, thus providing multiple instruction streams.

The interconnection network chosen for a parallel processing system will greatly influence the performance of the machine. Many questions

about the capabilities of interconnection networks remain unanswered. This is especially true for the SIMD system environment where the interconnection network will often be called upon to transfer information among all N processors simultaneously, that is, to perform data permutations. Poor performance in passing needed permutations could render a particular interconnection network unsuitable for use in an SIMD system by causing a serious degradation in processor utilization.

PASM, a partitionable SIMD/MIMD parallel processor, is a reconfigurable multimicroprocessor system under development at Purdue University. It is designed to operate in either mode of parallelism. Two interconnection networks are being considered for use in PASM: the Generalized Cube and the Augmented Data Manipulator (ADM). The Generalized Cube has been studied. Various properties of the ADM have been examined but further investigation is needed. This work is concerned with the capabilities of the ADM network in SIMD mode. Increased knowledge of ADM network performance will allow a more informed choice of an interconnection network for PASM.

The general model of SIMD parallel processing systems to be used throughout this work is described in Chapter 2. Chapter 3 contains a brief overview of PASM. In Chapter 4, the two networks are formally defined. The setting of the networks to perform permutations is discussed. Chapter 5 presents the argument for counting the number of distinct permutations passable by the Generalized Cube network, for later comparison with the ADM network.

Chapter 6 investigates one parameter of ADM network performance - the number of passable permutations. This development involves both

properties of network topology and combinatorial mathematical techniques. First, the number of permutations performable by a network stage is counted. Then, using network partitioning, the arguments are extended to provide upper and lower bounds on the number of data permutations passable by the ADM network. An exact count is given for the case N = 8. Finally, the asymptotic behavior of the bounds is analyzed and a comparison with the number of Generalized Cube performable permutations is made.

The use of routing tags for distributed network control is discussed in Chapter 7. Routing tag schemes for the Generalized Cube are The permuting capability of the network is not at all limited by the routing tag control.

Chapter 8 reviews two families of routing tags for the ADM network. A count of the number of permutations passable using positive dominant or negative dominant permutation routing tags is given. The number of permutations passable using natural permutation routing tags is bounded.

Algorithms are developed in Chapter 9 to determine the passability of an arbitrary permutation by the ADM network under distributed control by any of three related routing tags. The correctness of the algorithm is proven.

Further ADM network properties, derived using group theory, are presented in Chapter 10. The permutations passable by the ADM are additionally characterized as a result.

CHAPTER 2

MODEL OF SIMD MACHINES

The acronym SIMD stands for single instruction stream - multiple data stream [FL]. Typically, an SIMD machine is a computer system consisting of a control unit, N processors, N memory modules, and an interconnection network. The control unit broadcasts instructions to all of the processors, and all active processors execute the same instruction at the same time. Thus, there is a single instruction stream. Each active processor executes the instruction on data in its own associated memory module. Thus, there is a multiple data stream. The interconnection network, sometimes referred to as an alignment or permutation network, provides a communication facility for the processors and memory modules.

One way to model the physical structure of an SIMD machine is shown in Figure 2.1. As indicated, there are N processing elements (PEs), where each PE consists of a processor with its own memory. The PEs receive their instructions from the control unit. Communication among the PEs is accomplished through the use of the interconnection network. This structure is called the PE-to-PE approach. The Illiac IV [BOU] is an example of this configuration.

Because each processor has direct access to its local memory module and relatively poorer access to any other memory module, tasks requiring transfers of large blocks of data between PEs should be avoided.

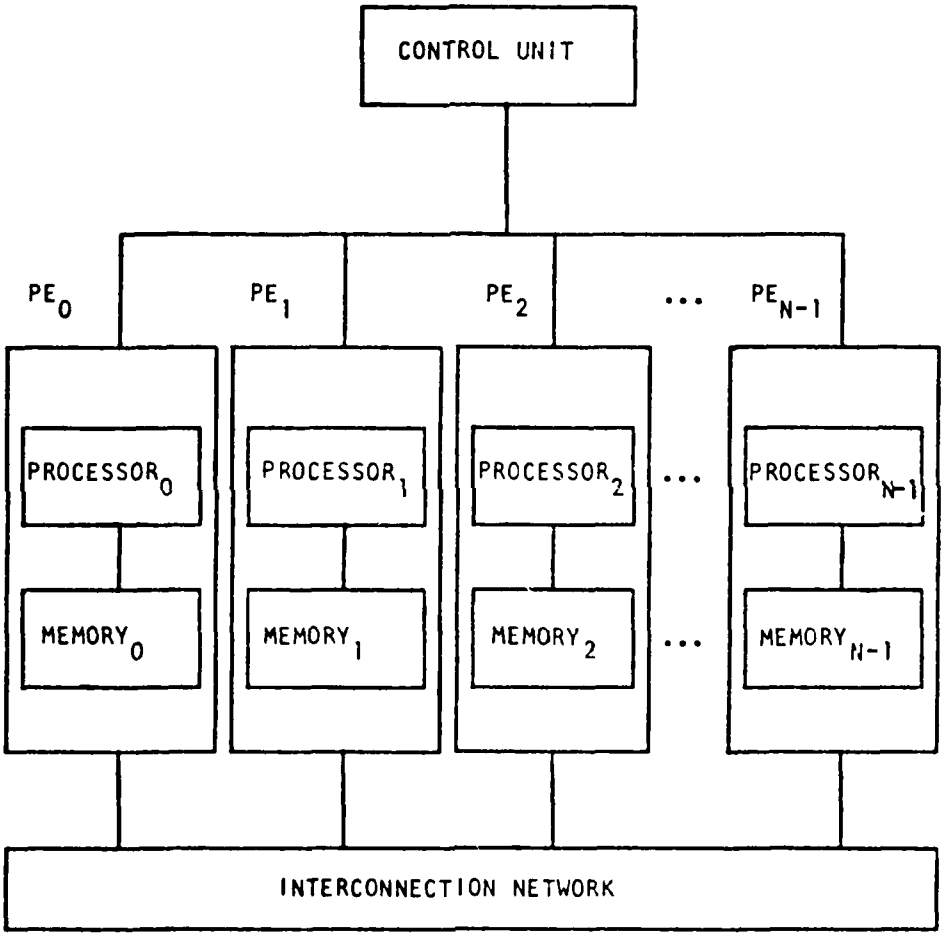


Figure 2.1 A PE-to-PE model of an SIMD machine.

Rather, algorithms involving a data base which can be partitioned into largely noninteracting segments are most suited to this structure. Communication between PEs can be supported by a unidirectional interconnection network since each PE has access to a network input and output.

A second way to model an SIMD machine is shown in Figure 2.2. This is the processor-to-memory approach. In general, there may be P processors connected to M memories through the interconnection network in this approach. The figure shows the case where there are N processors and N memories. The BSP [JE] is an SIMD machine with this structure.

In this case transfer of large blocks of data from processor to processor is easily accomplished by using the interconnection network to change the memory module linked to a given processor. One disadvantage of this architecture is that each instruction or data fetch must pass through the network. Another is that two processors can only communicate through a shared memory module.

For the processor-to-memory structure, processors must be able to perform memory read and write operations through the interconnection network. If the processors and memories have fixed access to the interconnection network, then it must support bidirectional communication. A unidirectional interconnection network can be used if provision is made to allow either processors or memories to be attached to both network inputs and outputs.

Further information about SIMD machine structures is contained in [ST]. Variations on the PE-to-PE and processor-to-memory architectures are discussed in [BA] and [LA]. A mathematical model of SIMD machines is presented in [SIS].

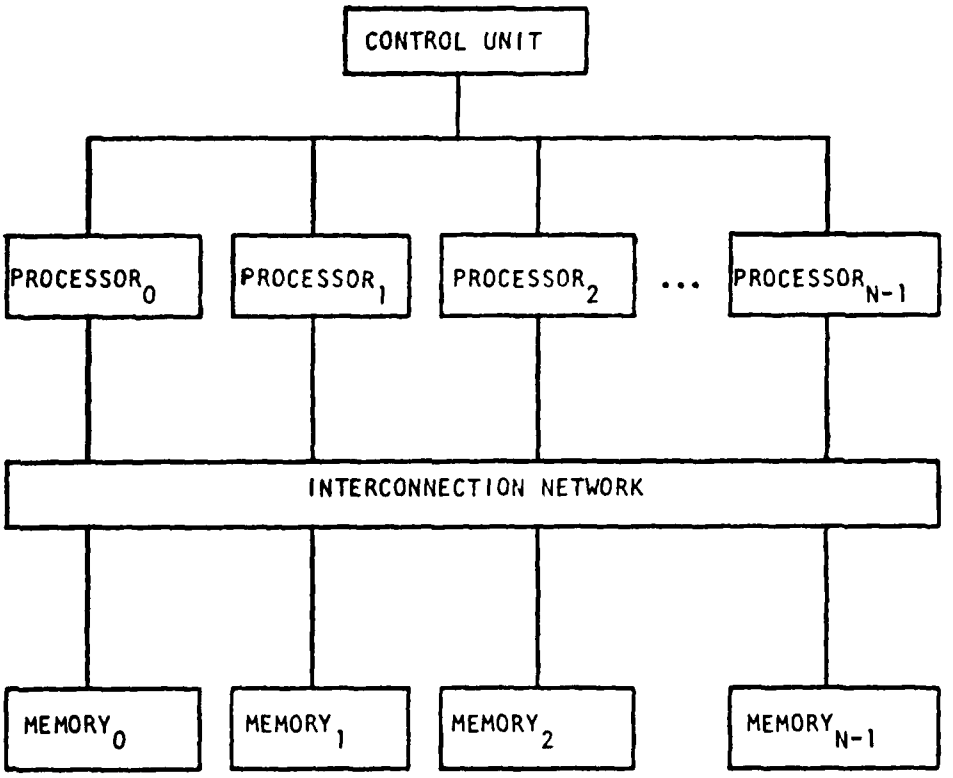


Figure 2.2 A processor-to-memory model of an SIMD machine.

The model of SIMD machines to be used in this and subsequent chapters, is the PE-to-PE model [SI5]. Each PE is assigned a unique address from 0 to N-1, represented in binary as $p_{n-1}p_{n-2}\cdots p_1p_0$. The results obtained for the ADM network will be valid for either model, however.

CHAPTER 3

OVERVIEW OF PASM

There are several types of parallel processing systems. An SIMD (single instruction stream - multiple data stream) machine typically consists of a set of N processors, N memories, an interconnection network, and a control unit (e.g. Illiac IV). The control unit broadcasts instructions to the processors and all active ("turned on") processors execute the same instruction at the same time. Each processor executes instructions using data taken from a memory with which only it is associated. The interconnection network allows interprocessor communication. An MIMD (multiple instruction stream - multiple data stream) machine usually consists of N processors and N memories, where each processor can follow an independent instruction stream (e.g. C.mmp). As with SIMD architectures, there is a multiple data stream and an interconnection network. A partitionable SIMD/MIMD system is a parallel processing system which can be structured as two or more independent SIMD and/or MIMD machines. In this chapter, the basic organization of PASM, a partitionable SIMD/MIMD system being designed at Purdue University for image processing and pattern recognition, is briefly overviewed.

SIMD machines can be used for "local" processing of segments of images in parallel. For example, the image can be segmented, and each processor assigned a segment. Then, following the same set of instructions, such tasks as line thinning, threshold dependent operations, and

gap filling can be done in parallel for all segments of the image simultaneously. Also in SIMD mode, matrix arithmetic used for such tasks as statistical pattern recognition can be done efficiently. MIMD machines can be used to perform different "global" pattern recognition tasks in parallel, using multiple copies of the image or one or more shared copies. For example, in cases where the goal is to locate two or more distinct objects in an image, each object can be assigned a processor or set of processors to search for it. An SIMD/MIMD application might involve using the same set of microprocessors for preprocessing an image in SIMD mode and then doing a pattern recognition task in MIMD mode.

PASM is a special purpose, dynamically reconfigurable, large-scale multimicroprocessor system. Due to the low cost of microprocessors, computer system designers have been considering various multimicrocomputer architectures. PASM was the first multimicroprocessor system in the literature to combine the following features: (1) it can be partitioned to operate as many independent SIMD and/or MIMD machines of varying sizes; and (2) a variety of problems in image processing and pattern recognition will be used to guide the design choices.

Figure 3.1 is a block diagram of the basic components of PASM. The System Control Unit (SCU) is a conventional machine, such as a PDP-11, and is responsible for the overall coordination of the activities of the other components of PASM. By carefully choosing which tasks should be assigned to the SCU and which should be assigned to other system components (such as the Memory Management System), the SCU can work effectively and not become a bottleneck.

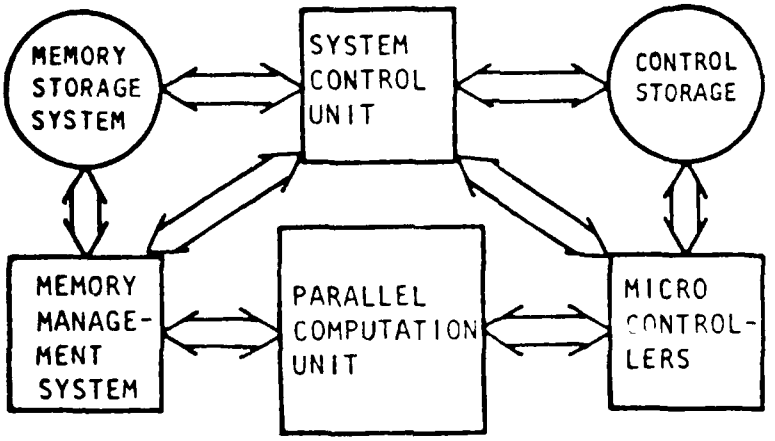


Figure 3.1 Block diagram overview of PASM.

The Parallel Computation Unit (PCU) contains $N = 2^n$ processors, N memory modules, and an interconnection network. The PCU processors are microprogrammable microprocessors that perform the actual SIMD and MIMD computations. The PCU memory modules are used by the PCU processors for data storage in SIMD mode and both data and instruction storage in MIMD mode. A memory module is connected to each processor to form a processor - memory pair called a processing element (PE) as shown in Figure 3.2. A pair of memory units is used for each memory module. This double-buffering scheme allows data to be moved between one memory unit and secondary storage (the Memory Storage System) while the processor operates on data in the other memory unit.

The interconnection network provides a means of communication among the PCU PEs. Two different interconnection networks are being considered for PASM: the Generalized Cube and the ADM. Both consist of $n = \log_2 N$ stages of switches and are controlled by routing tags. Both can be partitioned into independent subnetworks if all of the PEs in a partition of size $P = 2^p$ have the same value in the low order n-p bit positions of their addresses. Studies are currently being conducted to choose which of these networks to implement in PASM. This work is a part of that effort.

The Micro Controllers (MCs) are a set of $Q = 2^q$ microprogrammable microprocessors, numbered (addressed) from 0 to Q-1, which act as the control units for the PCU processors in SIMD mode and orchestrate the activities of the PCU processors in MIMD mode. Each MC is attached to a memory module (a pair of memory units so that memory loading and computations can be overlapped). Control Storage contains the programs for

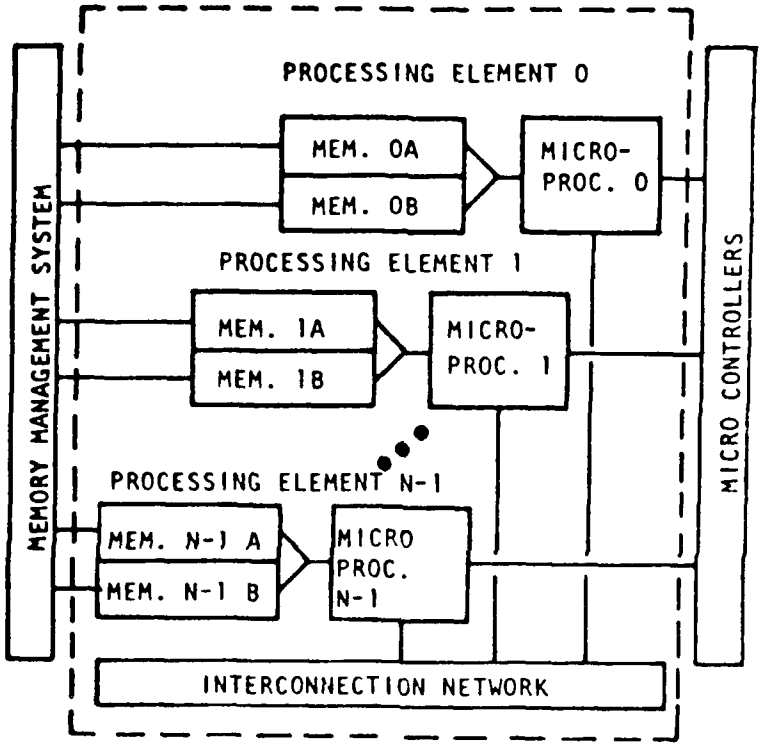


Figure 3.2 PASM Parallel Computation Unit.

the MCs.

Each MC controls N/Q PCU processors. The physical addresses of the N/Q PEs connected to an MC, shown in Figure 3.3, have as their low-order q bits the physical address of the MC, so that the network can be partitioned. Possible values for N and Q are 1024 and 16, respectively. A virtual SIMD machine (partition) of size RN/Q, $R = 2^r$ and $1 \leq r \leq q$, is obtained by loading R MC memory modules with the same instructions simultaneously. In SIMD mode, the R MCs are synchronized and each MC fetches instructions from its memory module, executing the control flow instructions (e.g. branches) and broadcasting the data processing instructions to its PCU PEs. Similarly, a virtual MIMD machine of size RN/Q is obtained by combining the efforts of the PCU processors of R MCs. In both cases, the physical addresses of these MCs must have the same low-order q-r bits so that all of the PCU PEs in the partition have the same low-order q-r physical address bits.

In each partition, the PCU PEs are assigned logical addresses. Given a virtual machine of size RN/Q, the PEs have logical numbers, 0 to (RN/Q)-1, (the high-order r+n-q bits of the physical number). Similarly, the MCs are assigned logical numbers from 0 to R-1 (for R > 1, the high-order r bits of its physical number). The PASM language compilers and operating system will be used to convert from logical to physical addresses, so a system user will deal only with logical addresses.

The Memory Management System controls the loading and unloading of the PCU memory modules. It employs a set of cooperating dedicated microprocessors. The Memory Storage System provides secondary storage for these files. Multiple devices are used to allow parallel data

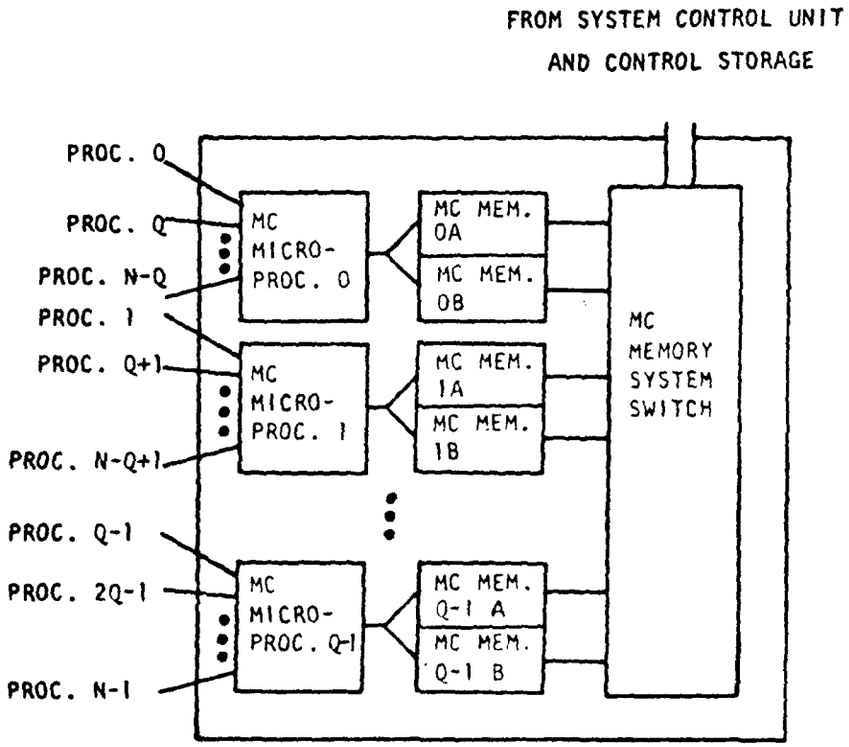


Figure 3.3 PASM Micro Controllers.

transfers.

The Memory Storage System will consist of N/Q independent Memory Storage units, numbered from 0 to (N/Q)-1. Each Memory Storage unit is connected to Q PCU memory units. For $0 \leq i < N/Q$, Memory Storage unit i is connected to those memory modules whose physical addresses are of the form $(Q*i)+k$, $0 \leq k < Q$. Thus, Memory Storage unit i is connected to the i^{th} processor/memory module pair of each MC as shown in Figure 3.4. Since the PE memories are double-buffered, while one job is being processed, results from the previous job can be stored and the next may be loaded.

The two main advantages of this approach for a partition of size N/Q are that (1) all of the memory modules can be loaded in parallel and (2) the data is directly available no matter which partition (MC group) is chosen. This is done by storing in Memory Storage unit i the data for a task which is to be loaded into the i^{th} logical memory module of the virtual machine of size N/Q, $0 \leq i < N/Q$. Thus, no matter which MC group of N/Q processors is chosen, the data from the i^{th} Memory Storage unit can be loaded into the i^{th} logical memory module of the virtual machine, for all i, $0 \leq i < N/Q$, simultaneously, i.e., in one parallel block transfer. This same approach can be taken if only $(N/Q)/2^d$ distinct Memory Storage units are available, $0 \leq d \leq n-1$, using 2^d parallel block loads will be required instead of just one. In general, a task needing RN/Q processors, $1 \leq R \leq Q$, logically numbered 0 to (RN/Q)-1, will require R parallel block loads if the data for the memory module whose high-order n-q logical address bits equal i is loaded into Memory Storage unit i. This is true no matter which group of R MCs (which

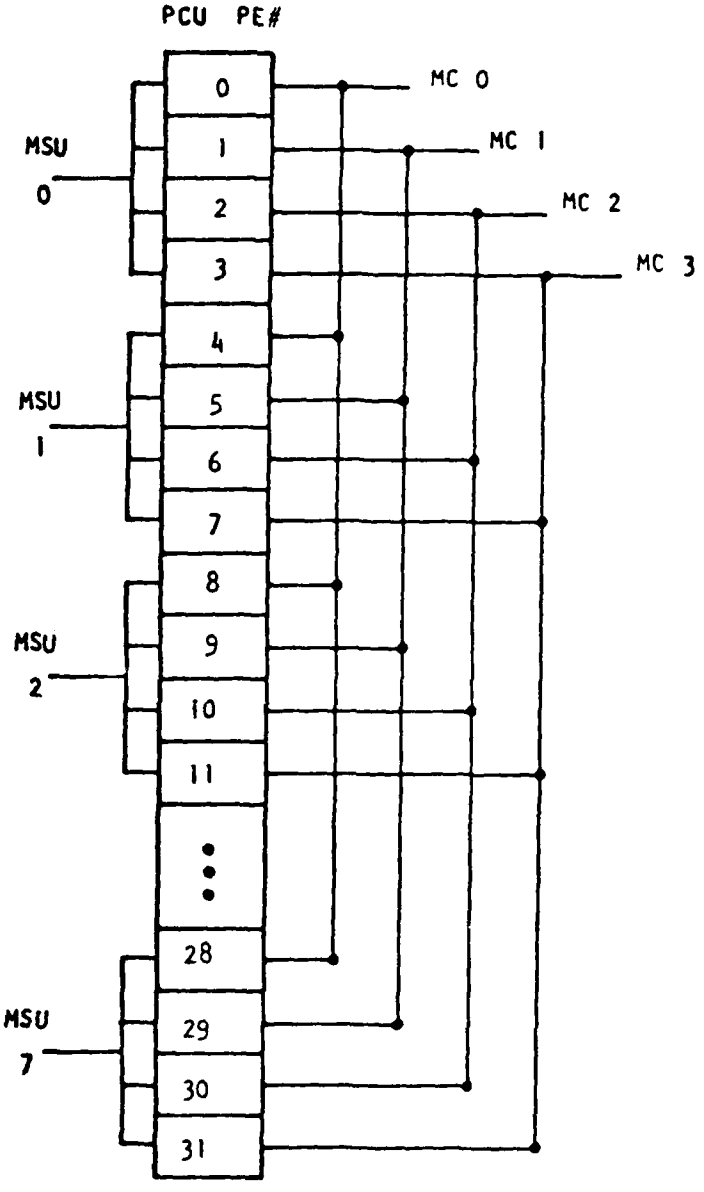


Figure 3.4 Organization of the PASM Memory Storage System for N = 32 and Q = 4, where "MSU" is Memory Storage Unit, "MC" is Micro Controller, and "PCU" is Parallel Computation Unit.

agree in their low-order q-r address bits) is chosen. If only $(N/Q)/2^d$ distinct Memory Storage units are available, $0 \leq d \leq n-q$, then $R*2^d$ parallel block loads will be required instead of just R.

A set of microprocessors is dedicated to performing the Memory Management System tasks in a distributed fashion, i.e., one processor handles Memory Storage System bus control, one handles the scheduling tasks, etc. This distributed processing approach is chosen in order to provide the Memory Management System with a large amount of processing power at low cost and high speed (due to the parallelism possible).

This overview of PASM, a large scale partitionable SIMD/MIMD multimicroprocessor system for image processing and pattern recognition, has been provided as background material for the following chapters. For additional information about various aspects of PASM see: organization [SI3,SMS1,SSKMS], instruction set [SM1], masking schemes for enabling and disabling PEs [SI1,SI2,SMS1,SSKMS], interconnection networks [MS,SI1,SI4,SI5,SI6,SS1,SS2,SSMA], operating system [SSMMS], programming language [MSS1], and memory management system [SKW,SSKMS], and examples of use [SI7,FSS,MSS2,SMS2,SSE].

iAPTER 4

NETWORK DEFINITIONS

In the SIMD environment it is useful to describe the interconnection network as a set of interconnection functions, where each is a permutation (bijection) on the set of PE addresses [SI1]. When interconnection function f is applied, network input i is connected to network output f(i) for all i, $0 \leq i < N$, simultaneously. That is, saying that the interconnection function maps the source address S to the destination address D is equivalent to saying the interconnection function causes data sent on the input line with address S to be routed to the output line with address D.

The physical structure of an interconnection network can be described by several parameters. A link or connection carries messages or data in the network between other network elements. A switching element selects the link or links over which messages or data will be sent through the network. A set of links connecting a network input, or source, to a network output, or destination, is called a route.

The Generalized Cube network [SS1] is shown in Figure 4.1 for N = 8, where N is the number of inputs to the network. It is an $n = \log_2 N$ stage network where each stage implements one of the cube interconnection functions [SS1]. The n cube functions are defined by

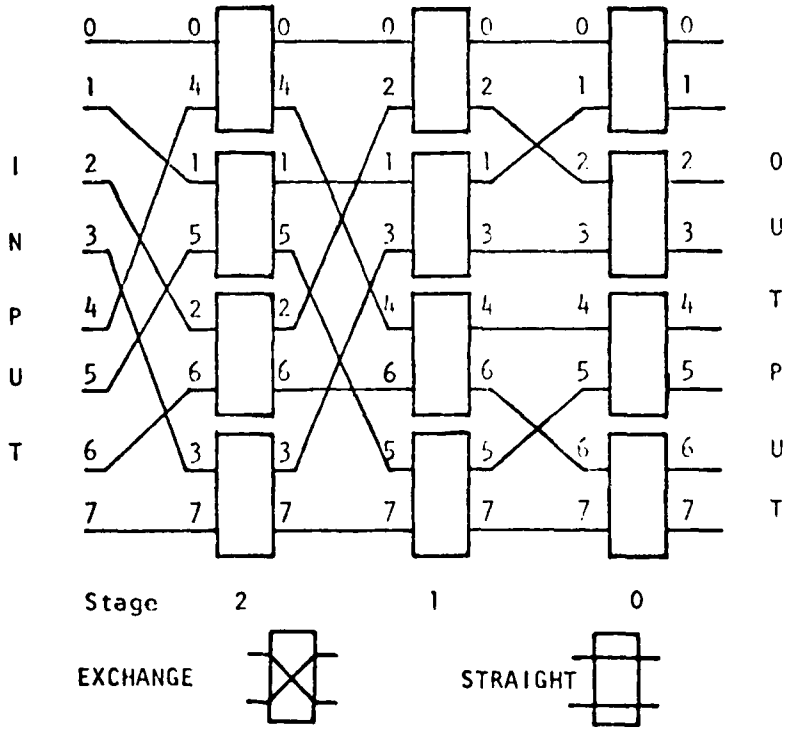


Figure 4.1 The Generalized Cube network for N = 8. The straight and exchange connections of the interchange box are shown.

$$cube_i(p_{n-1}\cdots p_0) = p_{n-1}\cdots p_{i+1}\bar{p}_i p_{i-1}\cdots p_0$$

for $0 \leq i < n$. The switching elements of this network are called interchange boxes. For performing permutations there are two legitimate states of an interchange box: (1) straight - input i to output i, input j to output j; and (2) exchange - input i to output j, input j to output i. In each stage of this network the pair of inputs to an interchange box is selected so that $cube_i$ maps one to the other, and vice versa. When an interchange box in stage i is set to exchange, the data items input to that interchange box are transferred as specified by the $cube_i$ interconnection function. When set to straight, data items input are transferred according to the identity function, i.e. identity $(p_{n-1}\cdots p_0) = p_{n-1}\cdots p_0$. Since each interchange box is individually controlled, each stage i will perform the $cube_i$ interconnection function on some subset of the data items depending on the settings of the interchange boxes.

There is a class of cube-type networks of which the Generalized Cube is representative. By combining the results of [SI4,SS1,WF1,WF2] it can be seen that all of the following networks are topologically equivalent: Generalized Cube [SS1], the STARAN flip network [BA], the omega network [LA], and the indirect binary n-cube network [PE]. (The SW-banyan (S=F=2) is defined as a graph [GL] and has the same topology as a multistage cube [WF2].) For this reason the Generalized Cube can be used as a standard for comparing cube-type networks with other interconnection networks.

The augmented data manipulator (ADM) network is shown in Figure 4.2 for N = 8. It is an N input, n stage network based on the PM2I (plus-

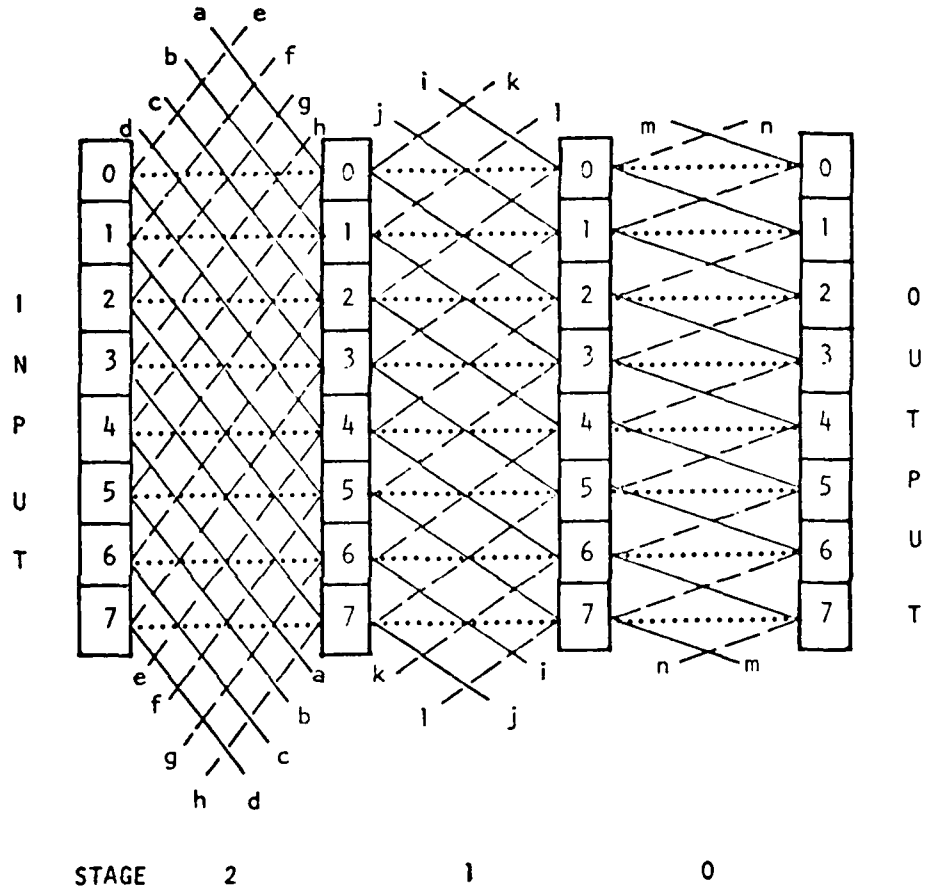


Figure 4.2 The augmented data manipulator (ADM) network for N = 8. Straight connections are shown by the dotted line; $PM2_{+i}$, by the solid lines; and $PM2_{-i}$, by the dashed lines.

minus 2^i) interconnection functions [SI1]. Each of the n stages consists of N switch cells. There is also an (n+1)-st column of network output cells. The PM2I functions are defined by

$$PM2_{+i}(j) = j + 2^i \text{ modulo } N$$

and

$$PM2_{-i}(j) = j - 2^i \text{ modulo } N$$

for $0 \leq j < N$, $0 \leq i < n$. Note that $PM2_{+(n-1)} = PM2_{-(n-1)}$. Each cell of the ADM can receive none, one, two, or three of the signals straight, $PM2_{+i}$, and $PM2_{-i}$ [SI1,SI6]. Corresponding to Figure 4.2, the signal "$PM2_{+i}$" means use the solid line connection; "$PM2_{-i}$," the dashed line connection; and "straight," the dotted line connection. Stages of the network are numbered from n-1 to 0. The data output from cell j at stage i becomes the data input to cell k at stage i-1 where $k \in \{j-2^i$ modulo N, j, $j+2^i$ modulo N$\}$. Each cell is controlled independently of any other cell.

The ADM network is based on Feng's data manipulator [FE]. The data manipulator is also based on the PM2I functions and consists of n+1 columns of N cells. There are again three connections from an input cell j at stage i, namely $PM2_{+i}$, $PM2_{-i}$, and straight, where $0 \leq j < N$ and $0 \leq i < n$. All but the last column are controlled by a pair of signals selected from a group of six. $U_1^{2^i}(PM2_{-i})$, $D_1^{2^i}(PM2_{+i})$, and $H_1^{2^i}$(straight) control those input cells at stage i whose i^{th} address bit is 0. The signals $U_2^{2^i}(PM2_{-i})$, $D_2^{2^i}(PM2_{+i})$, and $H_2^{2^i}$(straight) control those cells whose i^{th} address bit is 1. Thus, the ADM is a data

manipulator network with individual cell control.

In an SIMD environment, the network configuration established in the Generalized Cube or ADM network would depend on the permutation of network inputs to outputs desired. As an example, for the permutation which maps any input x to (x+3) modulo N, $0 \leq x < N$, the settings for both networks, when N = 8, are shown in Figures 4.3 and 4.4. Not all permutations of N items can be performed by these networks in one pass through the network. However, the permutation capability of the ADM network is known to be a superset of that of the Generalized Cube [SI4,SS1].

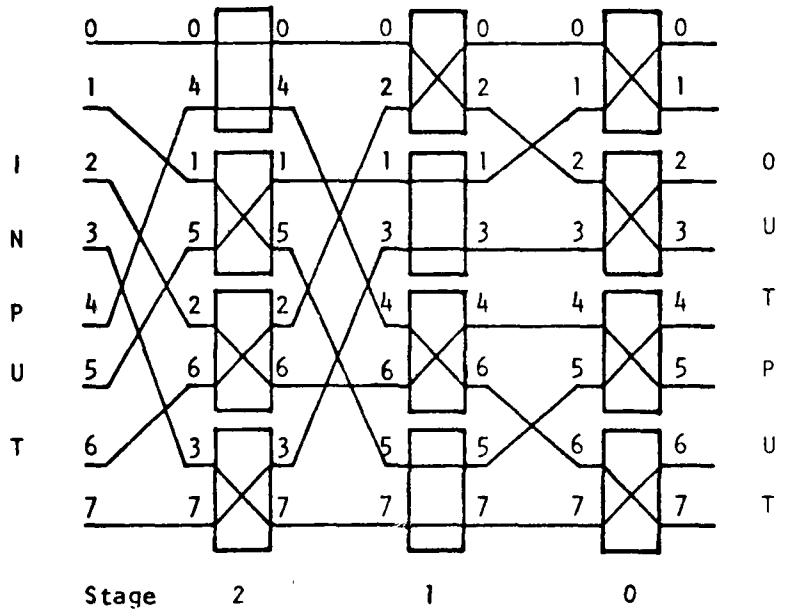


Figure 4.3 Interchange box settings for performing a cyclic shift of +3 modulo 8 in the Generalized Cube.

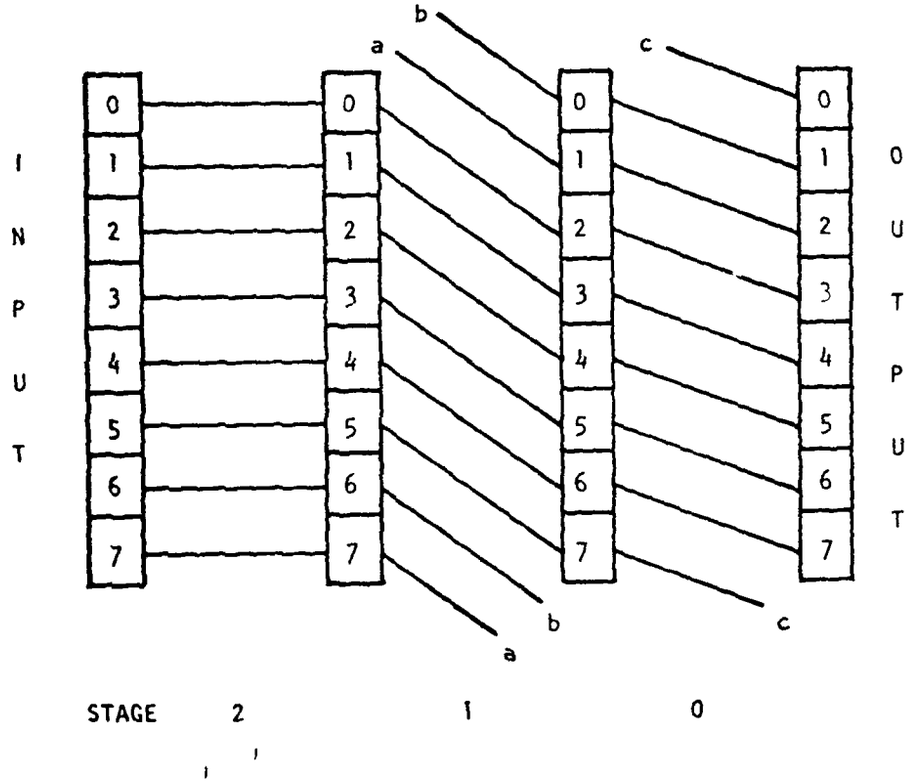


Figure 4.4 One possible ADM network setting for performing a cyclic shift of +3 modulo 8.

CHAPTER 5

COUNTING GENERALIZED CUBE PERMUTATIONS

The N-input Generalized Cube network has Nn/2 interchange boxes. For permuting data, each interchange box can be individually set to one of two states, either straight or exchange (see Figure 4.1). Thus, there are $2^{Nn/2}$ different ways to set the Nn/2 interchange boxes. It is clear from the structure of the network that every possible setting will result in a one to one mapping of inputs to outputs, i.e. a permutation, since each interchange box performs one to one connections.

The following theorem is needed to show that a one-to-one correspondence exists between network settings and permutations for the Generalized Cube.

Theorem 5.1: There is one and only one route between any source and destination for the Generalized Cube network.

Proof: Consider an arbitrary source, $S = (s_{n-1}...s_0)$, and a destination, $D = (d_{n-1}...d_0)$. For a route connecting S to D to exist, the cube_i interconnection functions, $0 \leq i < n$, which are implemented by the physical network hardware must be able to map S to D. In each stage of the network there is exactly one interchange box with an input labelled by some given address. Thus S can be mapped to D if first in stage n-1 the interchange box with S as an input is set to straight if $s_{n-1} = d_{n-1}$ or set to exchange if $s_{n-1} \neq d_{n-1}$. The straight connection maps

$(s_{n-1}\cdots s_0)$ to $(s_{n-1}\cdots s_0) = (d_{n-1}s_{n-2}\cdots s_0)$. The exchange connection performs $cube_{n-1}$ mapping $(s_{n-1}\cdots s_0)$ to $(\bar{s}_{n-1}\cdots s_0) = (d_{n-1}s_{n-2}\cdots s_0)$. This procedure can be repeated for stage n-2, setting the interchange box with $(d_{n-1}s_{n-2}\cdots s_0)$ as an input to the correct state to map $(d_{n-1}s_{n-2}\cdots s_0)$ to $(d_{n-1}d_{n-2}s_{n-3}\cdots s_0)$.

The procedure can be continued for stages n-3 through 0 mapping an arbitrary source to any destination. The procedure is deterministic and there is only one valid choice of interchange box state for each stage, so there is only one route between a source and destination.

[]

Now consider two distinct network settings. There must be at least one interchange box which is set straight in one of the settings, and exchange in the other. Pick a source, S, which is mapped to its destination, D, through this particular interchange box for one of the settings. There is only one path through the network between any source/destination pair. Thus, using the other setting does not allow S to map to D, giving a distinct permutation. A permutation is said to be passable by an interconnection network if the physical network structure (i.e., interchange boxes, for the Generalized Cube) allows the connections to be made. Therefore, each distinct setting results in a distinct permutation giving a total of $2^{Nn/2}$ permutations passable by the Generalized Cube (and its equivalents [SS1]).

This permutation count for the Generalized Cube network is relatively straightforward. It is included here for later comparison to the number of permutations performable by the ADM network.

CHAPTER 6

COUNTING AUGMENTED DATA MANIPULATOR PERMUTATIONS

6.1 Introduction

Unlike the case of the Generalized Cube network, the question of the number of distinct permutations passable by the ADM network does not yield to a straightforward consideration of all possible network states. There are two reasons for this difficulty. One is the fact that in the ADM, unlike the Generalized Cube, an arbitrary network setting may not result in a permutation of network inputs to outputs. Figure 6.1 shows an example of this. When the two routes of two different source/destination pairs have any links in common a collision is said to exist. Data passing through the network can be lost in this situation. In Figure 6.1 each cell is performing an allowable switch setting. However, in stage 1 both cells 1 and 4 connect to cell 1 and in stage 0 cell 5 connects to both cells 4 and 5. If the network setting is f, then f(1) = f(4) = 1 and f(5) = 4 or 5. Clearly, f is not a permutation. The second reason is that for certain permutations more than one valid network setting exists. Figure 6.2 shows two settings which are equivalent. In each case the same permutation of network inputs to outputs is performed, that is 0 to 3, 1 to 6, 2 to 5, 3 to 2, 4 to 7, 5 to 4, 6 to 1, and 7 to 0.

For the remainder of the discussion, ADM network performable permutations are referred to as overall permutations. Configurations of

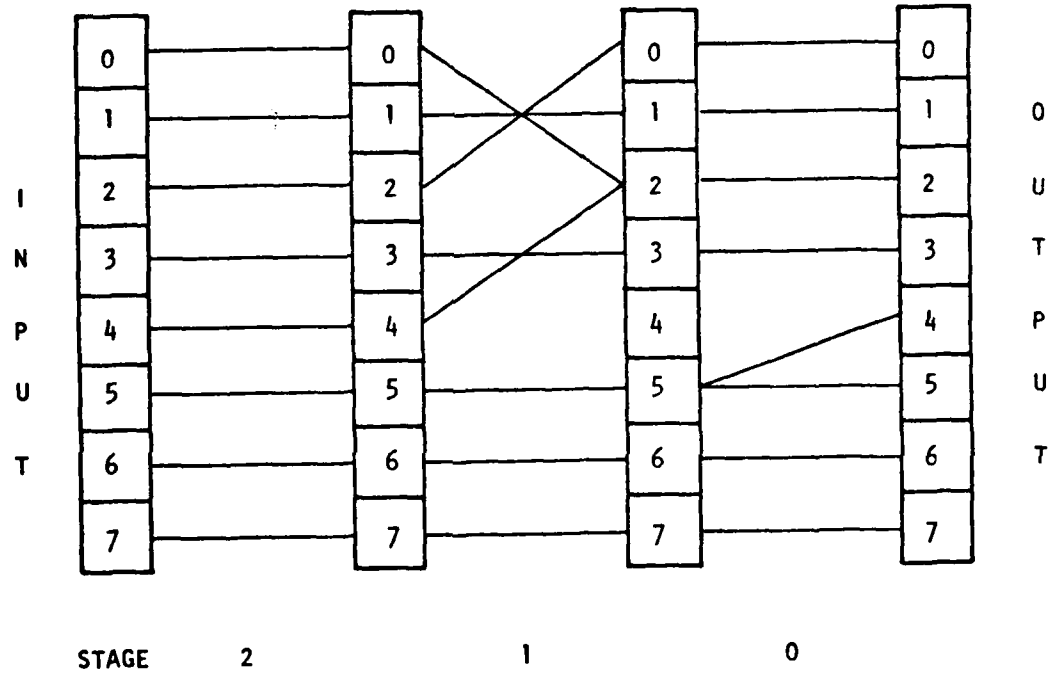


Figure 6.1 Example of a network setting for N = 8 which does not correspond to an overall permutation.

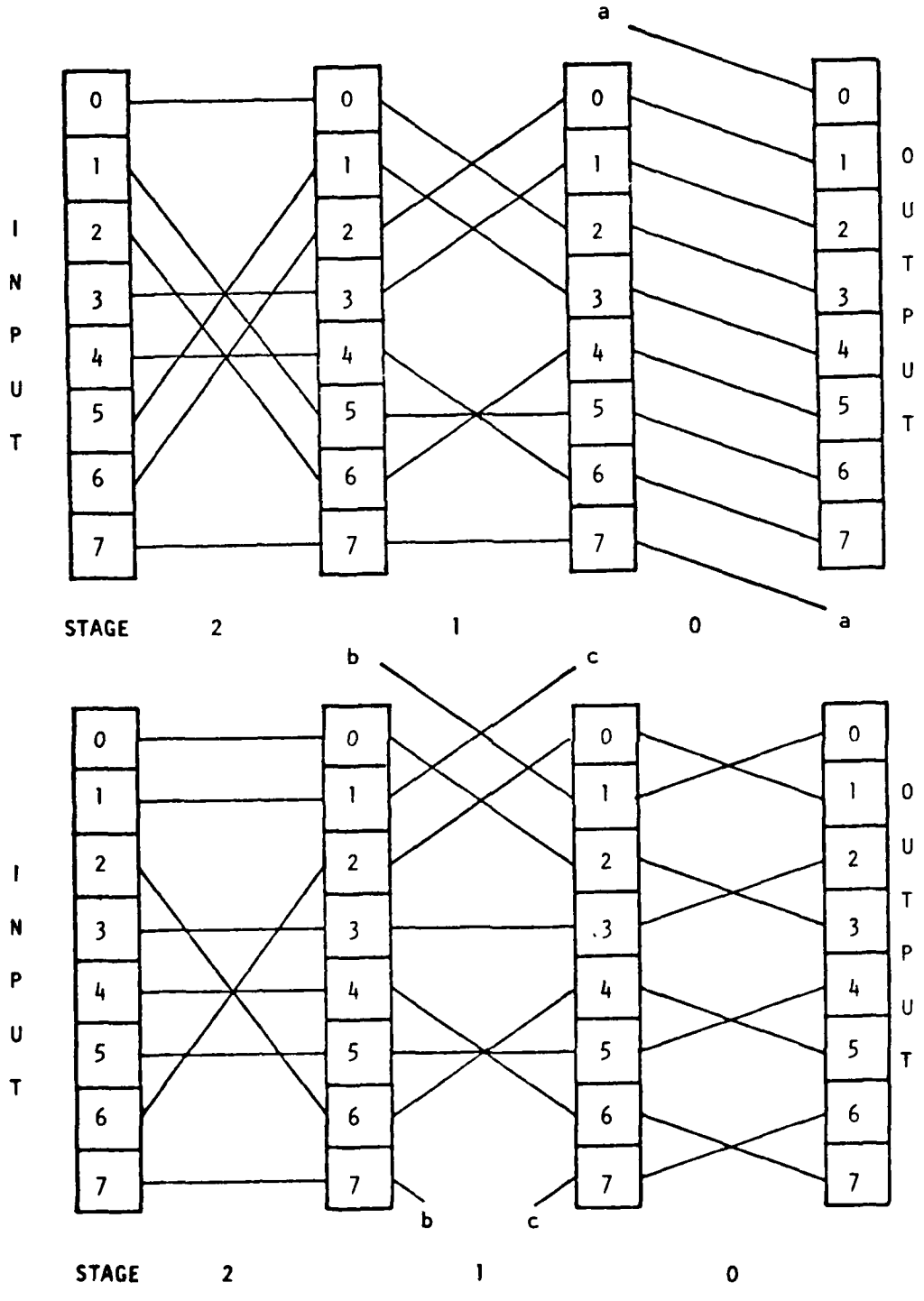


Figure 6.2 Example of two distinct network settings for N = 8 which correspond to the same overall permutation.

stage j of the network, for $0 \leq j < n$, which are permutations of stage j inputs to outputs are called stage j permutations.

The approach to counting the number of overall permutations will be first to determine what type of network settings give a permutation of inputs to outputs. Next, the number of stage 0 permutations is counted and the result generalized to any stage. Finally, using partitioning theory and the results concerning network stages, the network is treated as two subnetworks connected to stage 0, and upper and lower bounds on the number of data permutations performable by the entire ADM network are established.

6.2 Stage Permutations

Before the number of overall permutations performable by the ADM network can be counted, the two difficulties described in the previous section must be addressed. The following answers the first difficulty, that of determining which type of network settings correspond to permutations.

Lemma 6.1: An ADM network configuration is an overall permutation if and only if it consists of stage i permutations for all i, $0 \leq i < n$.

Proof: Assume a given network configuration is an overall permutation. For this to be true there can be no conflict of data at any cell in the network, i.e., no cell can receive data from more than one cell in the previous stage. Because each stage has the same number of cells, no cell can fail to receive data, without conflict or loss of data in that stage. Thus, if the network configuration is an overall permutation then each stage i configuration must be a permutation for all i,

$0 \leq i < n$.

Assume that each stage i configuration is a permutation for all i, $0 \leq i < n$. Because of this constraint on the stage configurations, no conflict can exist in the network. This implies that the network configuration is an overall permutation.

[]

This lemma, while obvious, is presented because it establishes the criteria for permutation passability in the ADM network network, which is central to the development that follows. A permutation is passable by the ADM network *if and only if a set of N routes exist which perform the desired mapping without conflict.*

To deal with the second difficulty, that of generating overall permutations with more than one network setting, a divide and conquer approach will be used. This will limit the need to check for setting redundancy to stage 0 of the network. First, the configurations of stage 0 are investigated.

Stage 0 is the only stage which can affect the low order bit of a source address, causing mapping to a destination with a low order bit that is either the same or different from that of the source. Let $S = (s_{n-1}s_{n-2}\cdots s_1 s_0)$ be a source, and $D = (d_{n-1}d_{n-2}\cdots d_1 d_0)$ its destination. A connection in stage 0 that does not affect the low order bit of the destination address, i.e., $s_0 = d_0$, is called a straight connection. A connection that changes the low order bit, $s_0 = \bar{d}_0$, is called an exchange. This is shown in Figure 6.3. A regular exchange is between stage 0 cells $(p_{n-1}\cdots p_1 0)$ and $(p_{n-1}\cdots p_1 0 + 2^0)$ modulo N. An irregular exchange is between stage 0 cells $(p_{n-1}\cdots p_1 0)$ and

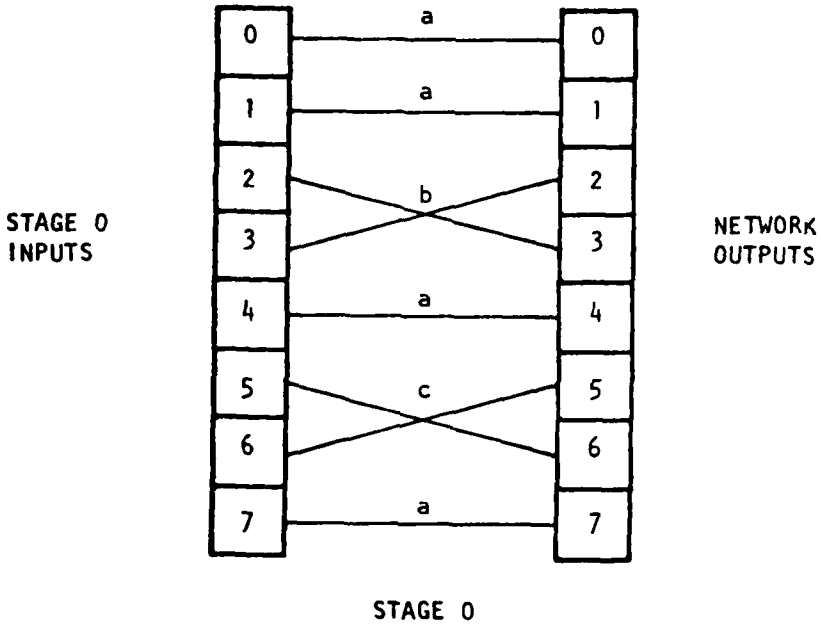


Figure 6.3 a) Straight connections, b) Regular exchange, c) Irregular exchange.

$(p_{n-1}\cdots p_1 0 - 2^0)$ modulo N. Because a permutation is one to one, any possible stage 0 permutation, except the all $+2^0$ or all -2^0 configurations, consists of straight and/or exchange connections only (i.e., every $+2^0$ or -2^0 connection is part of an exchange) [SI6]. The all $+2^0$ and all -2^0 connections form permutations because every cell uses $+2^0$ or every one uses -2^0 (modulo N arithmetic).

Consider the stage 0 permutations other than the all $+2^0$ or all -2^0. They can be represented by an N-bit binary number, called the characteristic binary number. A binary digit is associated with each adjacent pair of cells, including a digit for the wrap-around pairing of the cells labeled 0 and N-1. If the adjacent pair of cells together form an exchange connection, the characteristic binary digit is 1. If not, the digit is 0. An example of this assignment is shown in Figure 6.4.

In order to use the characteristic binary numbers for counting stage 0 permutations, two kinds of digit adjacency are distinguished. When the first and last bits of the characteristic binary numbers are not considered adjacent it is linear adjacency. When the first and last bits are considered adjacent it is circular adjacency.

Lemma 6.2: Every stage 0 permutation, except the settings all $+2^0$ or all -2^0, has a unique characteristic binary number with no circularly adjacent bits that are both 1s.

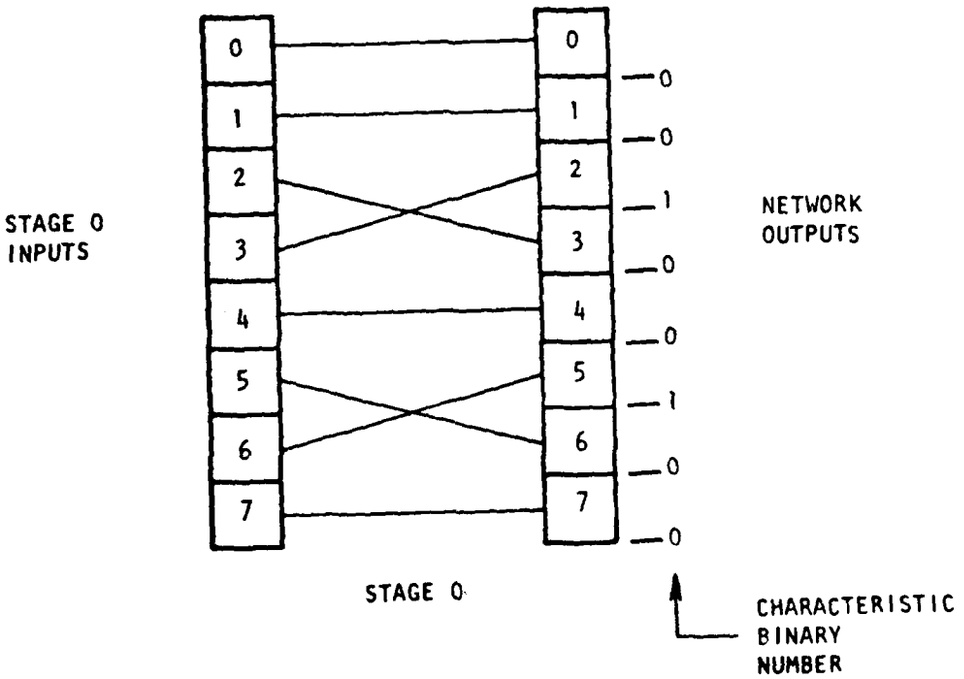


Figure 6.4 A stage 0 permutation and its characteristic binary number for N = 8.

Proof: Every stage 0 permutation, except the all $+2^0$ and all -2^0 configurations, can be formed from straight and exchange connections [SI6]. If the characteristic binary number of a configuration has circularly adjacent 1s, then there is a cell involved in two exchanges such that

$$p_{n-1}p_{n-2}\cdots p_1p_0 \rightarrow (p_{n-1}p_{n-2}\cdots p_1p_0+2^0) \text{ modulo } N$$

and

$$p_{n-1}p_{n-2}\cdots p_1p_0 \rightarrow (p_{n-1}p_{n-2}\cdots p_1p_0-2^0) \text{ modulo } N$$

This is shown in Figure 6.5. This mapping is not one-to-one, hence the configuration is not a permutation. If the associated binary number has no circularly adjacent 1s, then every stage 0 input can be involved in at most one exchange. Since every input is involved in either a straight or an exchange connection, the configuration will be one-to-one, and hence a permutation.

[]

The characteristic binary numbers of stage 0 permutations can be used to count the number of these permutations. Lemmas 6.3 and 6.4 are based upon [OS].

Lemma 6.3: The number of N-bit binary numbers with no linearly adjacent 1s is found using the recursive relationship

$$L(N) = L(N-1) + L(N-2)$$

where $L(2) = 3$, $L(3) = 5$, and $N \geq 4$.

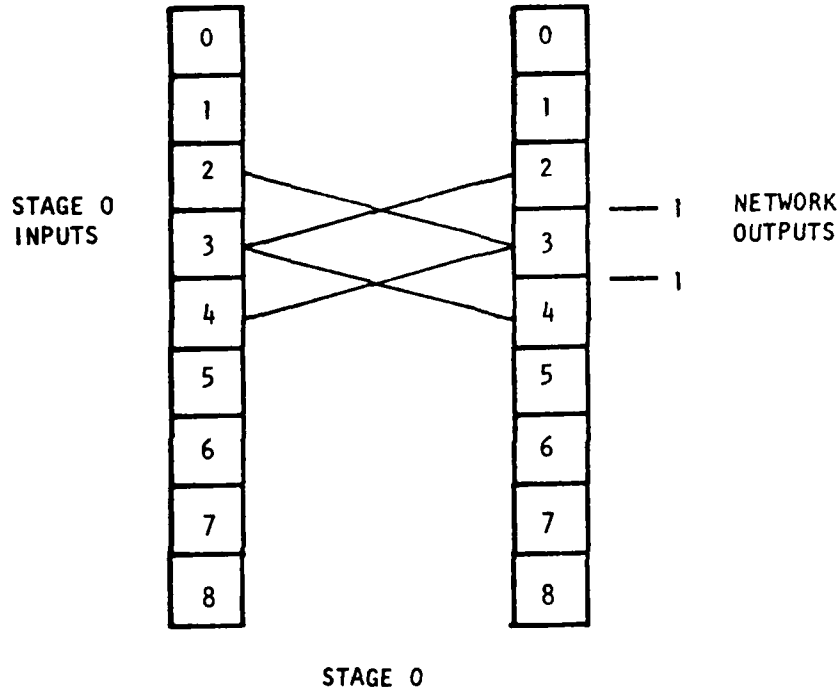


Figure 6.5 Configuration implied by two adjacent 1s in the characteristic binary number. This is not a permutation.

Proof: If an N-bit number ends in a 0, then it will have no linearly adjacent 1s if it has no linearly adjacent 1s in the first N-1 bits. The number of all such N-bit numbers is L(N-1). If an N-bit number ends in 1, then the immediately preceding bit must be a 0 if the number is to have no linearly adjacent 1s. Also, the first N-2 bits of the number must have no linearly adjacent 1s. The number of all such N-bit numbers is L(N-2). Thus, L(N) = L(N-1) + L(N-2).

The initial conditions may be derived by noting that L(2) = 3 (i.e.: 00,01,10) and L(3) = 5 (i.e.: 000,100,010,001,101).

[]

Lemma 6.4: The number of N-bit binary numbers with no circularly adjacent 1s is

$$C(N) = L(N) - L(N-4)$$

where $N \geq 8$.

Proof: L(N) exceeds C(N) by the number of N-bit numbers with no linearly adjacent 1s which do have circularly adjacent 1s. These numbers are all of the form

$$1\ 0\ a_1\ a_2\ \ldots\ a_{N-4}\ 0\ 1$$

where the number $a_1a_2\ldots a_{N-4}$ is a binary number with no linearly adjacent 1s. There are L(N-4) such numbers. Thus, C(N) = L(N) - L(N-4).

[]

Theorem 6.1: For an N-input ADM network, the number of stage 0 permutations is

$$P_0(N) = C(N) + 2$$

where $N \geq 8$. Also, $P_0(2) = 2$ and $P_0(4) = 9$.

Proof: By Lemma 6.2, $P_0(N)$ will be equal to the number of characteristic binary numbers with no circularly adjacent 1s, plus the two cases all $+2^0$ and all -2^0. $P_0(2)$ can be counted by direct enumeration. $P_0(4)$ can be counted either by direct enumeration or by noting that $C(4) = L(4) - 1$, the "-1" being for the case 1001.

[]

The method used to count the number of stage 0 permutations can be applied to any stage of the ADM network.

Theorem 6.2: For an N-input ADM network, the number of stage i permutations is

$$P_i(N) = P_0(N/2^i)2^i$$

where $N \geq 2$, and $0 \leq i < n$.

Proof: To count the number of ways in which stage i can permute data, consider the set of cells $S = \{j, j+2^i, j+2*2^i, \ldots j+(2^{n-i}-1)*2^i\}$ for a fixed j, $0 \leq j < 2^i$. In stage i an arbitrary cell, k, can be mapped to any of {$(k-2^i)$ modulo N, k, $(k+2^i)$ modulo N}. That is, if $k \in S$, for any fixed j, $0 \leq j < 2^i$, then $(k-2^i)$ modulo $N \in S$ and $(k+2^i)$ modulo $N \in S$. Since successive elements of S differ by 2^i, this mapping of k is completely analogous to that of k' mapping to

$\{(k'-2^0)$ modulo N, k', $(k'+2^0)$ modulo N$\}$ where $k' \in S' = \{0,1,\ldots,2^{n-1}-1\}$. This second mapping is that of stage 0 in an ADM network with 2^{n-i} inputs. Thus the cells of S can perform $P_0(2^{n-1}) = P_0(N/2^i)$ permutations. There are 2^i possible values of j, each defining a set S^j such that $S^j \bigcap S^k = \emptyset$ for $k \neq j$ and $0 \leq j,k < 2^i$. That is, the 2^i sets are disjoint, so the permutations performable on the cells of S^j are independent of those performable on S^k, for $k \neq j$. Thus $P_i(N) = P_0(N/2^i)^{2^i}$.

[]

6.3 Network Permutations

The remaining obstacle to counting the number of distinct ADM performable overall permutations is determining what class of permutations have more than one network setting. If the network is small this task can be avoided. For an ADM network where N = 4, the number of performable permutations can be counted by direct enumeration.

Lemma 6.5: For N = 4, the ADM network can perform all possible N! = 24 permutations.

Proof: By direct enumeration (see [SS3]).

[]

For N > 4, direct enumeration is not a practical alternative for counting the number of overall permutations. Consider conceptually separating stage 0 from the rest of the network. This is shown in Figure 6.6. Stages n-1 through 1 can be partitioned into two independent subnetworks, each with N/2 inputs [SI6,SS1]. All the odd-numbered cells

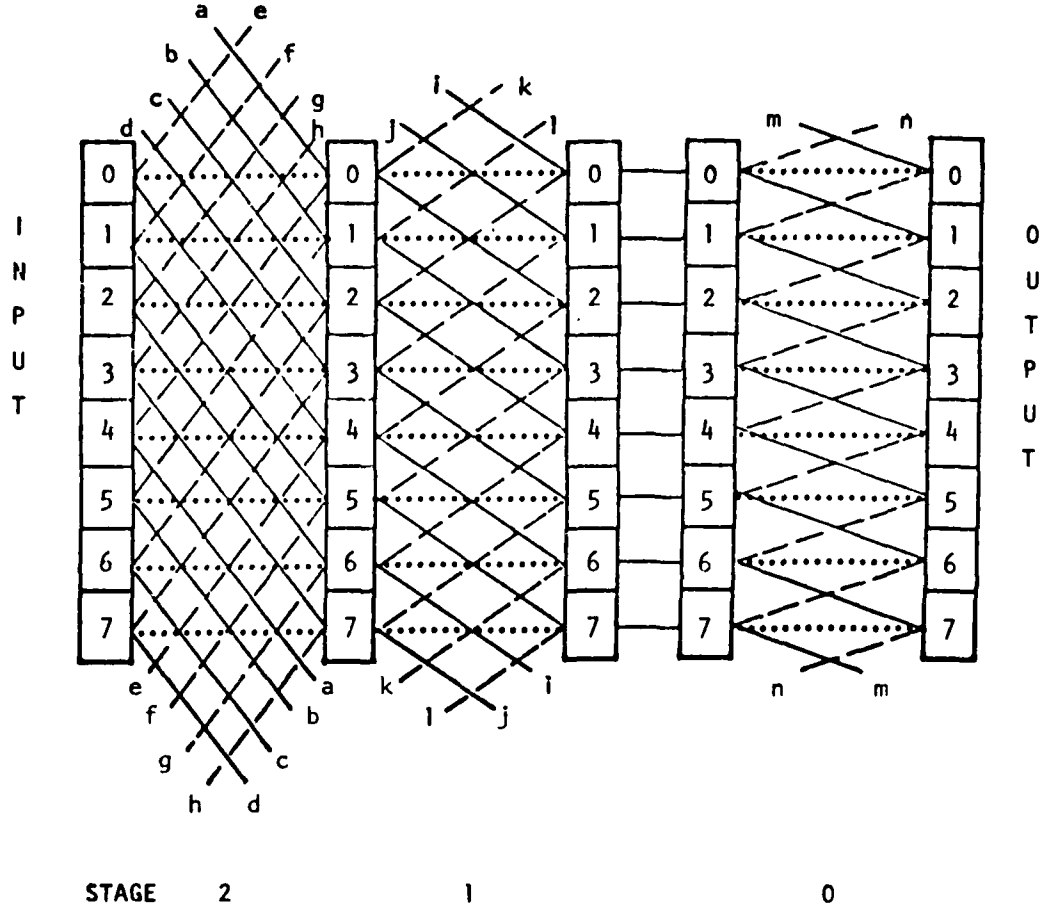


Figure 6.6 The first step in the conceptual process of partitioning an ADM network for N = 8 into two independent subnetworks joined at stage 0. Each cell that interfaces stage 1 and 0 is shown divided into an output cell from stage 1 and an input cell to stage 0.

in stages n-1 through 1 will constitute one of the subnetworks. This is the odd subnetwork. All the even-numbered cells in stages n-1 through 1 constitute the other subnetwork, the even subnetwork. The relationship of these two subnetworks to the final stage of the N-input ADM network, stage 0, is shown in Figure 6.7.

The partitioning described connects the outputs of the even subnetwork to all even-numbered inputs of stage 0. The outputs of the odd subnetwork are connected to the odd-numbered inputs of stage 0. Partitioning the ADM network allows an N-input network to be treated as two N/2-input independent ADM networks combined at stage 0 of the N-input network.

Lemma 6.6: The four stage 0 permutations all regular exchanges, all irregular exchanges, all $+2^0$, and all -2^0 connect all even subnetwork outputs to odd numbered network outputs and all odd subnetwork outputs to even numbered network outputs. Furthermore, no other stage 0 permutation does this.

Proof: The four named permutations each connect all even subnetwork outputs to odd network outputs and all odd subnetwork outputs to even network outputs because each forces $d_0 = \overline{s}_0$ for all source/destination pairs. This is shown in Figure 6.8 for N = 8. A permutation not of the named set must have a straight connection. If an output, D, is connected to a straight stage 0 link, then $d_0 = s_0$. Thus, the four named stage 0 permutations are the only ones with this property.

[]

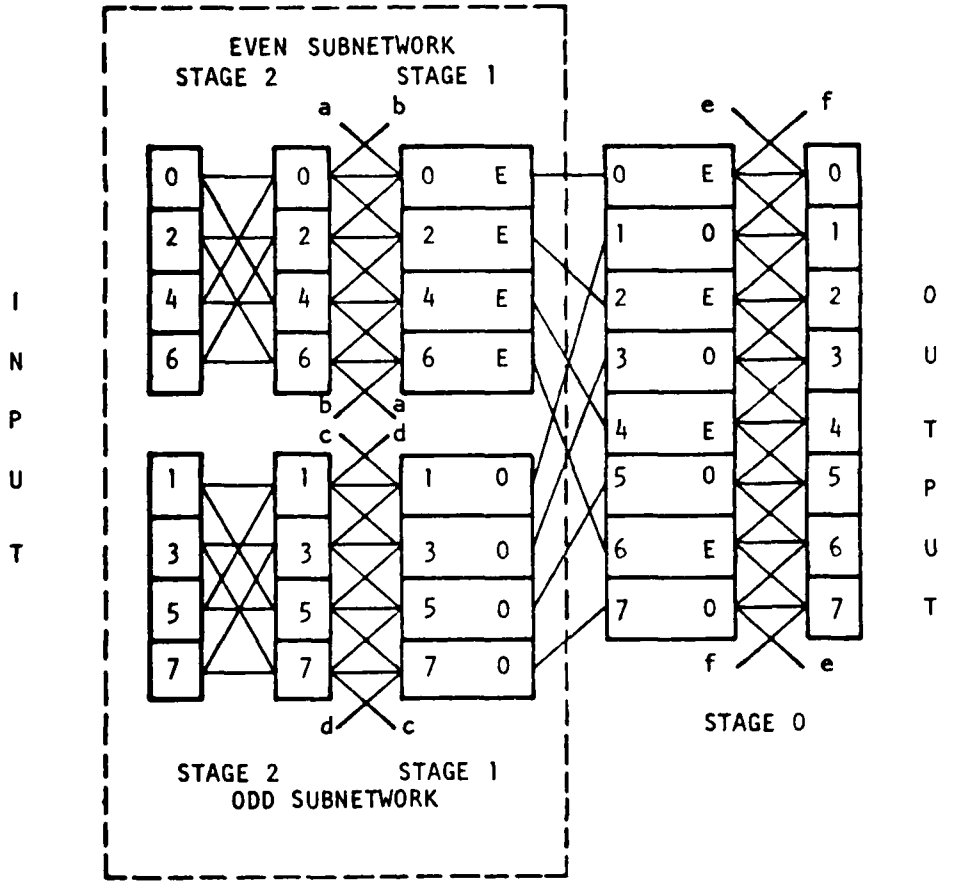


Figure 6.7 Cells from stages 2 and 1 of an ADM network for N = 8 rearranged into the two independent subnetworks, each with N/2 inputs. E and 0 designate even and odd subnetwork, respectively.

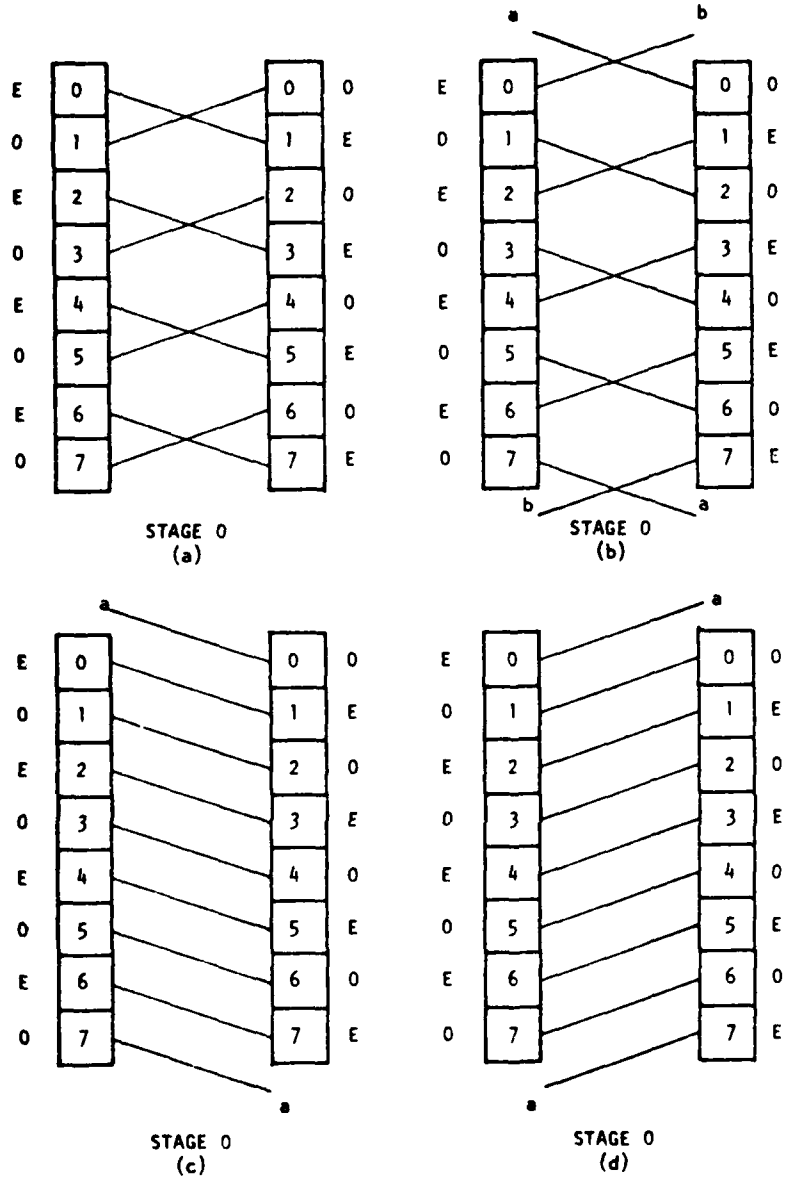


Figure 6.8 Illustration of the four stage 0 permutations for N = 8 which connect all even subnetwork outputs to odd network outputs, and odd subnetwork outputs to even network outputs. They are (a) all regular exchanges, (b) all irregular exchanges, (c) all $+2^0$, and (d) all -2^0. Even and odd source subnetworks are indicated by E and O, respectively.

Consider an arbitrary destination, D, of an N-input ADM network. The source for the data arriving at D may have been either the even subnetwork or the odd subnetwork (see Figure 6.7) depending on the stage 0 configuration. Call the subnetwork which is the source of D the source subnetwork.

Lemma 6.7: Consider the set of all stage 0 permutations except all regular exchanges, all irregular exchanges, all $+2^0$, and all -2^0. For each of these permutations the set of pairings of source subnetwork with network output is unique.

Proof: Proof by contradiction. Assume that two distinct stage 0 permutations of this set both link the same source subnetwork to a given network output, and that this is true for any network output. That is, the two permutations have identical sets of pairings. A permutation of the named set must have a straight connection. If an output, D, is connected to a straight stage 0 link, then $d_0 = s_0$. If it is connected to an exchange, then $d_0 = \overline{s}_0$, and the source subnetwork differs from the previous case. Thus all straight connections of one permutation must be duplicated in the other, and vice versa, if the source subnetworks are to be the same for each output.

A circularly adjacent pair of 0s in a characteristic binary number corresponds to a straight connection. Specifying all straight connections thus specifies all circularly adjacent 0s in the characteristic binary numbers of both permutations. The remaining bits of the numbers must contain no circularly adjacent 0s. Recall that no circularly adjacent 1s may appear since this is a permutation. Thus, the first and

last bits of any contiguous unspecified bit positions must be 1s and the interior bits must alternate 1s and 0s. Single unspecified bit positions must become 1s. Each unspecified bit position is thus assigned a unique value. Therefore, both numbers are identical. The permutations cannot be distinct.

[]

Let P(N) be the number of distinct overall permutations performable by an N-input ADM network. And let $P_U(N)$ and $P_L(N)$ be upper and lower bounds on P(N), respectively.

Theorem 6.3: A lower bound on the number of distinct overall permutations performable by the N-input ADM network is

$$P_L(N) = P_L(N/2)^2 * [P_0(N)-3]$$

where $P_L(4) = P(4) = 24$; $N \geq 8$.

Proof: As a result of Lemma 6.1 only configurations which are permutations at every stage need be considered in the following. The number of permutations performable by one of the two independent subnetworks which can be formed by partitioning the ADM is, by definition, at least $P_L(N/2)$. Call the permutations available at the inputs of stage 0 the input permutations. Because the two subnetworks of the partition are independent, the number of distinct input permutations is at least $P_L(N/2)^2$.

Now, consider an arbitrary overall permutation. Assume that the stage 0 permutation is fixed. Any change in the input permutation will result in a change in the overall permutation. This is because

($a \cdot c = b \cdot c$) implies $a = b$ where a, b, and c are permutations and $\cdot$ is composition. Assume the input permutation and the stage 0 permutation are both allowed to change. Let the stage 0 permutation be restricted so that only one of the permutations all regular exchanges, all irregular exchanges, all $+2^0$, or all -2^0 is allowed; there are $P_0(N)-3$ of these. As a consequence of Lemma 6.6 and Lemma 6.7, any change in the stage 0 permutation will cause at least one output address to be mapped from a different subnetwork. But because the two subnetworks are independent, no change of the input permutation can result in the mapping of a source of one subnetwork to the output of the other. The resulting overall permutation cannot be the same as the original no matter how stage 0 and the subnetworks are manipulated.

Thus, no overall permutation can be duplicated by changing the input permutation and/or changing the stage 0 permutation (provided the stage 0 permutation is not one of the three excluded). Hence, each composition of input permutation with allowable stage 0 permutation (i.e., not one of the three excluded) results in a unique overall permutation. So, the number of input permutations is multiplied by the number of stage 0 permutations, minus the three special cases, to yield the lower bound given on the number of performable overall permutations.

The boundary condition was stated in Lemma 6.5.

[]

Theorem 6.4: An upper bound on the number of distinct overall permutations performable by the N-input ADM network is

$$P_U(N) = P_U(N/2)^2 * P_0(N)$$

where $P_U(4) = P(4) = 24$ and $N \geq 8$.

Proof: Assuming that the composition of any input permutation with any stage 0 permutation (including all regular exchanges, all irregular exchanges, all $+2^0$, and all -2^0) yields a unique overall permutation, gives the above result.

[]

For an ADM network with N = 8, an exact count of the number of performable permutations can be derived.

Theorem 6.5: $P(8) = P(4)^2 * [P_0(8)-3]$.

Proof: From Lemma 6.6 and Lemma 6.7 the stage 0 permutations all regular exchanges, all irregular exchanges, all $+2^0$, and all -2^0 are the only stage 0 permutations which share a common set of pairings of source subnetwork with network output. Consider a particular overall permutation involving a stage 0 permutation selected from this set of four. The same overall permutation can be maintained after changing stage 0 to another of the given set of four stage 0 permutations if the input permutation can be suitably modified. For example, Figure 6.2 shows a given overall permutation in the upper network which uses the all $+2^0$ setting in stage 0. The lower network shows stage 0 set to all regular exchanges and the necessary changes in the settings of stages 2 and 1

made so that the same overall permutation is performed. The changes in stages 2 and 1 settings accomplish the needed input permutation modification. Since the choice of source subnetwork remains unchanged for all outputs after resetting stage 0, the necessary changes in the input permutation will occur only within the two independent stage 0 subnetworks. For N = 8 these subnetworks are themselves 4-input ADM networks which can perform any permutation of four items (Lemma 6.5). Therefore, any needed modification of the input permutation can be performed. So, the overall permutations performable using any member of the given set of four stage 0 permutations will be exactly the same as those performable using any of the three other stage 0 permutations. Thus P(N) is equal to the lower bound given in Theorem 6.3.

[]

Because the ADM network with N = 8 cannot perform all N! = 40,320 permutations (P(8) = 24,496), the method of Theorem 6.5 does not extend directly to larger values of N.

Corollary 6.1: The number of distinct permutations performable by the N-input ADM is bounded by

$$P_L(N/2)^2 * [P_0(N)-3] \leq P(N) < P_U(N/2)^2 * P_0(N)$$

where $N \geq 8$. Also

$$P_L(8) = P_U(8) = P(8) = 26,496.$$

Proof: This corollary follows from Theorems 6.3, 6.4, and 6.5. P(N) is strictly less than $P_U(N)$ because there exist overall permutations for which multiple distinct input permutation and stage 0 permutation compositions result in the same overall permutation. For example an overall permutation of input i to output (i+1) modulo N, $0 \leq i < N$, can be done with stage 0 set to all $+2^0$, all -2^0, or all regular exchanges.

[]

6.4 Tightness and Asymptotic Behavior of the Bounds

The suitability of the bounds given in Corollary 6.1 as a measure of ADM network performance will depend on how tight the bounds are for various values of N, or network size. The less the difference between the upper and lower bounds the more useful they are as an indicator of ADM network performance. The tightness of the bounds stated in Corollary 6.1 can be calculated as a function of the number of inputs to the network. Define the spread of the bounds, S(N), to be

$$S(N) = \frac{P_U(N) - P_L(N)}{P_L(N)} .$$

Using this formula and the results of Corollary 6.1 and Theorem 6.5, Table 6.1 is calculated. The number of permutations performable by the Generalized Cube network (see Chapter 5) is included in Table 6.1 for comparison.

Let $x = P_L(N_0)$ and $x+\Delta = P_U(N_0)$, where $N_0 = 2^i$ for $i \in \{0,1,2,\ldots\}$. Then for an ADM network with N_0 inputs the spread is

$$S(N_0) = \frac{P_U(N_0) - P_L(N_0)}{P_L(N_0)}$$

$$= \frac{(x+\Delta) - x}{x} .$$

Now the values of the lower and upper bounds for the next power of two larger ADM network are respectively $x^2 * [P_0(2*N_0) - 3]$ and $(x+\Delta)^2 * P_0(2*N_0)$. Using the approximation $P_0(N_0) \simeq P_0(N_0) - 3$ the spread is then

$$S(2*N_0) = \frac{(x+\Delta)^2 * P_0(2*N_0) - x^2 * P_0(2*N_0)}{x^2 * P_0(2*N_0)}$$

$$= \frac{(x+\Delta)^2 - x^2}{x^2} .$$

Since $P_0(32) = 4,870,849$ this is a good approximation for $N_0 \geq 32$. For an arbitrarily large ADM network the spread, using the approximation, is in general

$$S(N) = \frac{(x+\Delta)^{2^i} - x^{2^i}}{x^{2^i}}$$

where $N = 2^i * N_0$, and $i \in \{0,1,2,\ldots\}$. As network size increases without limit

$$\lim_{N\to\infty} S(N) = \lim_{2^i * N_0 \to \infty} S(N)$$

$$= \lim_{i\to\infty} S(N)$$

$$= \lim_{i\to\infty} \frac{(x+\Delta)^{2^i} - x^{2^i}}{x^{2^i}}$$

$$= \lim_{i\to\infty} \left[\frac{(x+\Delta)^{2^i}}{x^{2^i}} - 1 \right]$$

$$= \infty .$$

Thus the bound becomes a less precise measure of the capability of an ADM network as network size increases. However, as shown in Table 6.1, the value of S(N) is small for networks for considerable size. So, for practical values of N, the bounds given in Corollary 6.1 give a useful approximation of the number of ADM network performable overall permutations.

Table 6.1 $P_L(N)$ and $P_U(N)$ are the lower and upper bounds on the number of permutations performable by an N-input ADM network, respectively. S(N) is the spread of the bounds calculated as $[P_U(N)-P_L(N)]/P_L(N)$. Cube (N) is the number of permutations performable by an N-input Generalized Cube network, given by $2^{Nn/2}$.

N	$P_L(N)$	$P_U(N)$	S(N)	Cube (N)
4	24	24	0	16
8	26,496	26,496	0	4096
16	1.55×10^{12}	1.55×10^{12}	1.36×10^{-3}	4.29×10^{9}
32	1.17×10^{31}	1.17×10^{31}	2.74×10^{-3}	1.21×10^{24}
64	3.24×10^{75}	3.26×10^{75}	5.45×10^{-3}	6.28×10^{57}
128	5.90×10^{177}	5.97×10^{177}	1.09×10^{-2}	7.27×10^{134}
256	1.01×10^{409}	1.13×10^{409}	2.20×10^{-2}	1.80×10^{308}
512	1.22×10^{925}	1.28×10^{925}	4.44×10^{-2}	3.74×10^{693}
1024	1.50×10^{1957}	1.64×10^{1957}	9.09×10^{-2}	1.88×10^{1541}
2048	2.27×10^{4128}	2.70×10^{4128}	1.90×10^{-1}	6.34×10^{2585}

6.5 Conclusions

The type of interconnection network chosen for an SIMD machine will have far reaching consequences for the ultimate system performance or lack of it. Comparison of various candidate interconnection networks can involve many factors such as cost, partitionability, etc.

This chapter has considered the number of permutations performable by the ADM network. For the ADM network, counting the number of performable permutations is made difficult by the fact that the network has settings which do not yield a permutation. Also, the network has multiple settings for certain passable permutations.

A method was given for counting the number of stage i configurations which are permutations, for any size ADM network, $0 \leq i < n$. Using partitioning theory in a divide and conquer approach led to an upper and lower bound on the number of distinct overall permutations which an ADM network can perform. To assess the characteristics of the bounds their tightness and asymptotic behavior was investigated. For the special case $N = 8$, an exact count of the number of distinct overall permutations performable was proven. Finally, a comparison of the number of distinct permutations performable by the ADM and Generalized Cube networks was made.

CHAPTER 7

GENERALIZED CUBE PERFORMANCE WITH ROUTING TAGS

7.1 Introduction

Another measure of the utility of a particular interconnection network is its ability to operate without centralized control. For SIMD machines with a large number of processing elements, centralized control of the interconnection network may cause that component of the system to become a bottleneck.

One way to distribute control of the interconnection network among the N PEs is to use routing tags. Each PE first computes a routing tag for the data item it will send through the network. Then at each switching cell, logic circuits, capable of using the information of the routing tag to control the cell setting, select an appropriate path so the data item will reach the desired destination. With this scheme the overhead time needed to establish network settings is independent of network size. Therefore, an important consideration when comparing the relative merits of various interconnection networks is the nature and capabilities of the routing tags compatible with each design.

This chapter considers the operation of the Generalized Cube network using routing tags. A representative tag scheme is chosen and network performance studied. The results obtained are used for comparison with the ADM network.

7.2 Routing Tag Operation

There is only one route between any source and destination in the Generalized Cube network (see Theorem 5.1). Consider a routing tag scheme for the network which correctly specifies the one route for any source/destination pair. Such a routing tag scheme will generate the correct set of N routes for any permutation which is passable by the Generalized Cube. Thus, the full permuting capabilities of the network are available with such routing tags.

Several routing tag schemes exist which give the correct route for any source/destination pair. One possibility is to generate a tag, T, according to $T = (t_{n-1} \ldots t_0) = S \oplus D$ [MCM]. The tag is interpreted in the following way. If $t_i = 0$ then the interchange box with the input $(d_{n-1} \ldots d_{i+1} s_i \ldots s_0)$ is set to straight. This is the interchange box in stage i through which S is mapped to D (see Theorem 5.1). If $t_i = 1$, the interchange box is set to exchange. For example if S = 0101 and D = 1001 then T = 1100 and the interchange box settings are exchange, exchange, straight, and straight.

This routing tag scheme uses easily computed tags of n bits. Because the exclusive-or operation is commutative, the tap mapping D to S is the same as that for S to D. This allows handshaking to be performed easily, if desired.

Other routing tag schemes are possible. In [LA], a 2n bit routing tag consisting of the n-bit source and destination addresses is described which is also suitable for use with the Generalized Cube network. With these tags a processing element receiving data can compare its address with that given in the tag to detect network errors. Anoth-

er scheme allowing certain kinds of broadcasting is presented in [MCM,WE].

7.3 Conclusions

The Generalized Cube network is well suited for distributed control using routing tags. Any routing tag scheme which can specify the route between any source/destination pair allows unrestricted use of the permuting capabilities of the network.

CHAPTER 8

AUGMENTED DATA MANIPULATOR PERFORMANCE WITH ROUTING TAGS

8.1 Introduction

The importance of distributed network control was discussed in Section 7.1. In this chapter various routing tag schemes for controlling the ADM network are reviewed. These schemes were presented in [MS,SSMA].

One family of routing tags discussed does not allow unrestricted operation of the ADM network. However, these tags have the compelling characteristic of easy compatibility. Thus, more precise knowledge of the performance of these tags in an SIMD environment will aid in evaluating their feasibility, as well as that of the ADM network.

8.2 Routing Tag Schemes

The routing tag schemes discussed in this chapter are defined in [MS]. To characterize an arbitrary path in the ADM network a full routing tag is required. A full routing tag requires 2n bits and is of the form $F = (f_{2n-1}f_{2n-2}\cdots f_1 f_0)$. It can be defined such that the even numbered bits represent the magnitude of the route to be taken within a particular stage and the odd numbered bits represent the sign. That is, if $f_{2i} = 0$, the straight link in stage i is used regardless of the value of f_{2i+1}. If $f_{2i} = 1$ and $f_{2i+1} = 0$, the $+2^i$ link is used. If $f_{2i} = 1$ and $f_{2i+1} = 1$, the -2^i link is used. Control of the ADM network is

distributed by constructing each cell with sufficient logic to examine bits f_{2i+1} and f_{2i} of a routing tag and make the appropriate link selection. For example, with N = 16, and if the source is 5 and the destination is 12, one possible value for F is 10001110. The route taken is $+2^3$, straight, -2^1, $+2^0$. The use of full routing tags allows the ADM to perform any passable permutation. However, no function or algorithm of reasonable complexity is known which will give a set of N non-conflicting tags for all permutation passable by the ADM network. So, less flexible, but easily computed, routing tag schemes have been developed.

If all the sign bits in a full routing tag are the same, the information contained in those bits can be represented by one bit which is the sign bit for the whole tag. An n+1 bit routing tag, T, can be formed by computing the signed magnitude difference between the destination, D, and the source, S, such that $T = (t_n t_{n-1} \cdots t_1 t_0) = D - S = (d_{n-1} \cdots d_0) - (s_{n-1} \cdots s_0)$. The sign bit is t_n, where $t_n = 0$ indicates positive and $t_n = 1$ indicates negative. Bits $t_{n-1} \cdots t_0$ equal the magnitude or absolute value of D-S. If all N routing tags for a permutation are calculated in this way, then the permutation is said to be routed using natural permutation routing tags [SM2]. A natural routing tag consisting of only straight or $+2^i$-type connections is said to be positive dominant. A routing tag with only straight or -2^i-type connections is negative dominant. A given natural routing tag must be either positive or negative dominant.

To execute this scheme bits t_n and t_i are examined to determine the link to be used at stage i. If $t_i = 0$, the straight link is used re-

gardless of the value of t_n. If $t_i = 1$ and $t_n = 0$, the $+2^i$ link is used, and if $t_i = 1$ and $t_n = 1$, the -2^i link is used. With N = 16, and if the source and destination are again 5 and 12, respectively, then T = 12-5 = 00111. The route taken is straight, $+2^2$, $+2^1$, $+2^0$.

Clearly, for any arbitrary source/destination pair there exists a corresponding natural routing tag and a tag-specified connection path through the ADM network. Also, natural tags are easily computed. These properties indicate that natural routing tags may be suitable for control of the ADM network in an MIMD environment [MS].

In [MS] it is also shown that a natural routing tag and its two's complement are equivalent in that they both route the same source to the same destination. Letting T' be the two's complement of T, then from the previous example T' = 11001. Input 5 is still connected to output 12 but the route taken is -2^3, straight, straight, -2^0.

As a consequence of its definition, any natural routing tag must be either positive or negative dominant (this is determined by the value of t_n). The two's complement of any tag must have the opposite of the sign bit of the original unless T is zero, in which case T' = T. However, the all zero tag is both positive and negative dominant. Thus a positive dominant routing tag and a negative dominant routing tag must exist for any source/destination pair. So positive dominant and negative dominant routing tags can be used in an MIMD environment.

In an SIMD environment, however, the desirable characteristics of a routing tag scheme are more restrictive. For good utility routing tags should, in addition to requiring minimal computation for their generation, enable passage of needed overall permutations. That is, ideally,

N non-conflicting paths should be specified when a network performable permutation is needed. If all N tags are calculated as natural routing tags, then the permutation is said to be routed using natural permutation routing tags. A permutation is said to be routed using positive dominant permutation routing tags if those tags that are negative dominant in the set of natural permutation routing tags are converted to positive dominant. Negative dominant permutation routing tags are defined correspondingly.

For most cases the ADM network provides two or more distinct routes between a source and its destination. Only when a source and destination have the same address is there only one route (for a proof of this see Lemma 10.1). The n+1 bit routing tag scheme described above can specify at most two distinct routes between a source/destination pair - one positive dominant and one negative dominant. If more routes exist they cannot be exploited with this scheme. However, the ease of generating these tags may make them useful in many instances, despite their limitations.

8.3 The Number of Permutations Passable Using Positive or Negative Dominant Tags

In Chapter 6 partitioning theory was applied to the ADM network at stage 0 to give two independent subnetworks on stages n-1 through 1 (see Figures 6.6 and 6.7). That is the output cells of these subnetworks are stage 1 output cells. The subnetworks so created have the structure of N/2-input ADM networks. Because of this each of the subnetworks may in turn be partitioned in the same manner as the whole network. This can be seen as partitioning the entire network at stage 1. Two independent

subnetworks will be created on stages n-1 through 2 from each of the subnetworks on stages n-1 through 1. This gives four independent subnetworks whose output cells are stage 2 output cells and which each have the structure of an N/4-input ADM network.

This process may be repeated until N/2 independent subnetworks whose output cells are stage n-1 output cells are generated. These subnetworks have the structure of a 2-input ADM network. At each stage i there are 2^i independent subnetworks created by this process. Summing over all stages gives the total number of subnetworks generated as

$$\sum_{i=0}^{n-1} 2^i = 2^0 * \left(\frac{1-2^n}{1-2}\right)$$

$$= 2^n - 1$$

$$= N-1.$$

Note that by including the term for i = 0 in the summation, the entire network has been considered a subnetwork of itself.

Since each subnetwork has the structure of an ADM network then it has the equivalent of a stage 0. This stage 0 will consist of all stage i input and output cells and their straight, $PM2_{+i}$, and $PM2_{-i}$ links contained in a given subnetwork which is created by the partitioning process at stage i. This set of cells and links is called the subnetwork stage 0.

The number of overall permutations passable by the ADM network using positive dominant permutation routing tags can now be counted.

Theorem 8.1: The ADM network can pass 2^{N-1} distinct overall permutations using positive dominant permutation routing tags.

Proof: Consider stage 0 of the entire network. Any exchange connection involves the use of $+2^0$ and -2^0 links. The -2^0 link cannot be used by positive dominant tags, so no exchanges can be performed in stage 0 when positive dominant tags are used. For the same reason, the all -2^0 setting cannot be performed. This leaves only the all straight and all $+2^0$ settings, which can be performed with positive dominant tags.

For each of the subnetworks the same reasoning applies. The subnetwork stage 0 for each subnetwork may be set only to all straight or all $+2^i$. However, each subnetwork stage 0 may be set independently because the subnetworks are independent (this includes stage 0 of the entire network). Since there are N-1 subnetworks there are 2^{N-1} distinct settings of the entire network.

Since the setting of each subnetwork stage 0 is a permutation, the setting for the entire network performs a permutation. Consider again stage 0 of the entire network. The all straight and all $+2^0$ settings do not have the same pairings of source subnetwork with network outputs (see Lemmas 6.6 and 6.7). The concept of source subnetwork can be extended naturally to each of the subnetworks created by the partitioning process by recalling that any subnetwork has the structure of an ADM network with the appropriate number of inputs. So in an analogous manner, the all straight and all $+2^i$ settings on each subnetwork stage 0 will not have the same pairings of source subnetwork with subnetwork outputs.

Thus, any change in a subnetwork stage 0 setting will cause at least one subnetwork output cell to be mapped from a different source subnetwork. The source subnetworks for a given subnetwork are independent. So changing a subnetwork stage 0 setting must give a different permutation. Thus the 2^{N-1} settings correspond to 2^{N-1} distinct passable overall permutations.

[]

Corollary 8.1: The ADM can pass 2^{N-1} distinct overall permutations using negative dominant permutation routing tags.

Proof: The proof is the same as for Theorem 8.1 except that the possible subnetwork stage 0 settings are all straight or all -2^i.

[]

When natural permutation routing tags are used additional permutations can be performed on the ADM network. The perfect shuffle is an example of one such permutation [MAS].

The number of permutation performable using natural tags can be bounded by considering the following. Some of the $PM2_{+i}$ links of stage i of the ADM network connect a stage i input cell j to a stage i output cell k where $j > k$. Also there are $PM2_{-i}$ links which connect l to m where $l < m$. These type connections are called wraparound connections. In Figure 4.2 these links are the ones drawn in two parts indicated by letters.

Lemma 8.1: ADM network wraparound connections are not used when the network is controlled using natural routing tags.

Proof: By definition, a natural routing tag, T, is computed as $T = D-S$, where S and D represent the source and destination addresses, respectively. If $S \leq D$, then $T \geq 0$ and the route which will be used is positive dominant. This means that at any stage i in the network data routed from S to D will pass through cells with addresses A_i such that $S \leq A_i \leq D$, for $0 \leq i < n$. When $S > D$, A_i is such that $S \geq A_i \geq D$, for $0 \leq i < n$. Thus no wraparound connections are used.

[]

The number of overall permutations performable without using the wraparound connections can be counted. Let $P_W(N)$ be the number of permutations performable by an N-input ADM network without using wraparound connections.

Theorem 8.2: Without using wraparound connections the number of distinct overall permutations the ADM network can perform is

$$P_W(N) = P_W(N/2)^2 * L(N-1)$$

where $P_W(2) = 2$.

Proof: The stage 0 permutations all irregular exchanges, all $+2^0$, and all -2^0 cannot be performed without using wraparound connections. Also, without wraparound all stage 0 permutations can be represented by a characteristic binary number which need not include a bit to represent the connections possible between cells 0 and N-1. Further, any

characteristic binary number with no linearly adjacent 1s will correspond to a unique stage 0 permutation. Deleting the bit for wraparound connections gives an N-1 bit characteristic binary number of which L(N-1) have no linearly adjacent 1s. There are two independent source subnetworks for stage 0. Each must not use any wraparound connections if the entire network is not to use any. Since each subnetwork has the structure of an N/2-input ADM network, each can perform $P_2(N/2)$ permutations without using any wraparound connections. Thus, the number of input permutations is $P_W(N/2)^2$. Because the allowable stage 0 permutations each have different pairings of source subnetworks with network outputs, $P_W(N) = P_W(N/2)^2 * L(N-1)$.

The value for $P_W(2)$ is found by direct enumeration.

[]

Let $P_{Nat}(N)$ be the number of permutations performable by an N-input ADM network using natural permutation routing tags.

Theorem 8.3: The number of permutations performable by the N-input ADM network using natural permutation routing tags is bounded by

$$P_{Nat}(N/2)^2 < P_{Nat}(N) < P_W(N)$$

where $P_{Nat}(2) = 2$.

Proof: By Lemma 8.1, natural routing tags do not use wraparound connections. Thus, by Theorem 8.2, $P_{Nat} \leq P_W(N)$. The inequality is strictly less than because there are permutations passable without using any wraparound connections but which cannot be done using natural routing tag routes. Figure 8.1 shows an example of this.

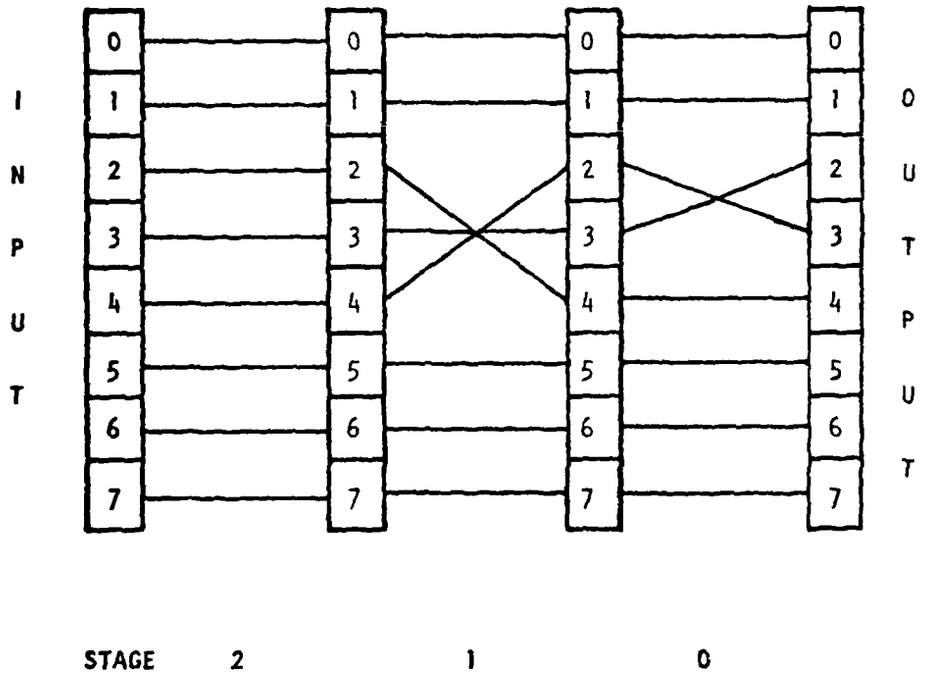


Figure 8.1 An overall permutation for N = 8 using no wraparound connections that is not performable using natural routing tags.

The lower bound can be achieved by setting stage 0 to all straight. There are then $P_{Nat}(N/2)^2$ input permutations using only natural permutation routing tags. However, other settings for stage 0 may be possible depending on the input permutation. If stages n-1 through 1 are each set to all straight, then any stage 0 setting not using wraparound connections would give a network setting which consisted of natural routes. By Lemma 6.7, any two stage 0 settings not using the wraparound connections (i.e., not all irregular exchanges, all $+2^i$, or all -2^i) will have different pairings of source subnetworks with network outputs. Thus changing the stage 0 setting will give a distinct overall permutation. So $P_{Nat}(N/2)^2$ is strictly less than $P_{Nat}(N)$.

[]

8.4 Conclusions

Routing tags for the ADM network were discussed. Full routing tags allow any route to be specified. Natural, positive dominant, and negative dominant permutation routing tags, while unable to specify arbitrary routes, are more easily computed than full tags, in general.

The number of overall permutations passable by positive and negative dominant tags was proven. The number of overall permutations passable using natural tags was bounded.

CHAPTER 9

ALGORITHMS FOR DETERMINING PERMUTATION PASSABILITY ON THE AUGMENTED DATA MANIPULATOR

9.1 Introduction

Chapter 8 discussed some of the limitations of natural, positive dominant, and negative dominant permutation routing tags. A count of the number of permutations passable using either positive or negative dominant tags was given, and the number performable using natural permutation routing tags was bounded. However, these results do not identify the overall permutations passable using these tags. The next section presents an algorithm which can be used to determine if a given arbitrary overall permutation is passable using natural, positive dominant, or negative dominant permutation routing tags.

9.2 The Algorithms

The procedure to check the passability of a given overall permutation using any one of the routing tag schemes discussed in Section 8.2 is given in Algorithm 9.1. In the procedure, the variable dest is an N-element vector where dest(j) is the destination associated with source j, for $0 \leq j < N$. The notation X_i is used to denote bit i of X.

To understand the operation of the algorithm, first recall that only stage 0 can affect the least significant bit of a source address. If $s_0 = d_0$ then stage 0 must be set to the straight link at the output cell with the destination address. If $s_0 \neq d_0$ then stage 0 must be set

```
1   procedure PASSABILITY(dest,N);
2   passable = true;
3   while passable = true do
4         for i = 0 step 1 until log2N-2 do
5             check = 0; /*check is an N-element vector*/
6                 for j = 0 step 1 until N-1 do
7                     if dest(j)i ≠ ji
8                         then REQUEST;
9                 if check(dest(j)) = 0
10                        then check(dest(j)) = 1;
11                        else passable = false;
12                  if passable = false then stop;
```

Algorithm 9.1 Procedure PASSABILITY

to a PM2I-type link at that cell.

Because natural routing tags specify a single specific path from a source to its destination, at each stage of the network there is only one cell through which a data item can pass. The N routing tags of the type selected must require that each data item pass through a distinct cell at each stage, if a permutation is to be passable using these tags. This is the criteria used by the algorithm to decide passability (see Lemma 6.1).

Before proceeding with the analysis of the algorithm it is useful to define two terms. Switching cells which select the connections to be used in stage i are called stage i input cells. Cells which are linked to by stage i input cells are stage i output cells. Note that stage i+1 output cells are stage i input cells.

The algorithm begins by comparing bits s_0 and d_0 of a source, j, and its destination, dest(j), where $0 \leq j < N$. If $s_0 = d_0$ a straight connection must be used to link a stage 0 input cell to the stage 0 output cell whose address is that of the destination. The address of the stage 0 input cell is then dest(j). A straight link in stage 0 is selected by retaining the current dest(j) value as the updated dest(j). This is because the data item must pass through the stage 0 input cell specified by the updated value of dest(j).

If $s_0 \neq d_0$ then a PM2I-type connection must be used to link a stage 0 input cell to the correct output cell. In this case the address of the stage 0 input cell will be either $(dest(j) + 2^0)$ modulo N or $(dest(j) - 2^0)$ modulo N, where the -2^0 and $+2^0$ links are used, respectively.

Positive dominant routing tags can only use the $+2^i$ links of the PM2I-type connections. Negative dominant routing tags can only use the -2^i links. So, updating dest(j) to the two possible stage 0 input cell address values serves to specify the $+2^0$ or -2^0 link. Updating dest(j) proceeds according to the instruction of REQUEST, three variations of which are listed in Algorithm 9.2. A different version of REQUEST exists for each type of tag that may be desired.

Consider the version for positive dominant permutation routing tags. When $s_0 \neq d_0$ the value of dest(j) is replaced by (dest(j) - 2^0) modulo N which specifies the $+2^0$ link. Again, the data item must pass through the stage 0 input cell specified by the updated value of dest(j).

After each dest(j) is updated, the N-element vector, check, which was initialized to all zeros, is used to indicate which stage 1 output cell (same as stage 0 input cell) has been selected by the new dest(j). The element check(dest(j)) is examined to see if it is a zero. If so, it is set to 1 to indicate that a data item will pass through this particular stage 1 output cell. If, however the element of check has already been set to 1 this indicates a conflict of data items and the given permutation is not passable using positive dominant permutation routing tags. The logic variable, passable, is set to false in this circumstance, and the algorithm terminates.

If bits s_0 and d_0 have been compared for each source/destination pair, and each dest(j) has been updated without conflict, then the check vector is reinitialized and the procedure begins anew for stage 1. This time bits s_1 and d_1 of each source/updated-destination pair are compared

For positive dominant routing tags, REQUEST is:

$dest(j) = (dest(j)-2^i)$ modulo N

For negative dominant routing tags, REQUEST is:

$dest(j) = (dest(j)+2^i)$ modulo N

For natural routing tags, REQUEST is:

if $(dest(j)-j) > 0$ then $dest(j) = (dest(j)-2^i)$ modulo N

else $dest(j) = (dest(j)+2^i)$ modulo N

Algorithm 9.2: Versions of REQUEST.

to determine the connections to use in stage 1. Since the setting of stage 0 has been established at this point, only stage 1 can have any affect on these bits.

The procedure is repeated for successive stages until either a conflict is detected, in which case the permutation is not passable and the algorithm terminates, or the connections in stage $n-2$ were found to be conflict free, implying the permutation is passable.

Stage $n-1$ need not be checked. To see why stage $n-1$ need not be checked, consider the case where the permutation under test is conflict free in stages 0 through $n-2$.

Since an overall permutation is being tested, if it is passable the stage $n-1$ configuration must be a stage $n-1$ permutation, as a consequence of Lemma 6.1. Recalled from Chapter 4 that $PM2_{+(n-1)} = PM2_{-(n-1)}$. That is each cell, A, can only be mapped to itself or $A+2^{n-1}$ modulo $N = A-2^{n-1}$ modulo N. So, a stage $n-1$ configuration is a stage n-1 permutation if and only if either the identity permutation or an exchange is performed on each pair of cells, A and $A+2^{n-1}$ modulo N, for $0 \leq A < N/2$. The identity permutation can be performed on any of the pairs of cells using routing tags of either dominance. The exchange permutation can be performed on any of the pairs of cells using positive dominant tags because A maps to $A+2^{n-1}$ modulo N via a $+2^{n-1}$ link and the $+2^{n-1}$ link from cell $A+2^{n-1}$ modulo N will map that cell to $(A+2^{n-1}) + 2^{n-1}$ modulo $N = A+N$ modulo $N = A$, completing the exchange. Note that negative dominant tags can perform these two permutations on any of the pairs of cells as well.

Connections in stage n-1 which are not stage n-1 permutations are not needed if the permutation under test is conflict free though stage n-2. There are two non-permutation settings possible on a pair of cells A and $A+2^{n-1}$ modulo N in stage n-1 which involve a single link from each cell. One connects A and $A+2^{n-1}$ modulo N to stage n-1 output cell A; the other connects the two cells to $A+2^{N-1}$ modulo N. If the algorithm checked stage n-1 it would specify one of these two connections only if dest(A) = dest($A+2^{n-1}$ modulo N), which cannot occur when there is no conflict through stage n-2. Settings involving none, two, or three links from either A or $A+2^{n-1}$ modulo N in stage n-1 are not needed because positive dominant routing tags (as well as negative dominant routing tags) define a single route from a source to destination.

Connections not performable by stage n-1, for example, A maps to stage n-1 output cell B where $B \neq A$ or $A+2^{n-1}$ modulo N, are also not needed to complete the network setting. After checking stage n-2, the algorithm will have updated the original values of dest(A) so that bits 0 through n-2 of A and dest(A) are the same. Thus, either A = dest(A) or they differ by 2^{n-1}. In either case, existing links in stage n-1 can perform the connection. Thus only a stage n-1 configuration which is a permutation will be required. Since the needed stage n-1 permutation can always be performed, that stage need not be checked.

The algorithm can be shown to give a correct result by demonstrating that it imitates the subtraction performed to calculate a routing tag. Expanding on the notation where X_i denotes bit i of X, let $X_{i/j}$, for $i \geq j$ represent bits i through j, inclusive, of X, and let $X_{i/j} = 0$

for $i < j$. First consider the case where positive dominant permutation routing tags are desired.

Theorem 9.1: Let D be a particular destination, S its source, and $R^{(i)}$ be the revised value of "dest(j)" after the inner loop of the algorithm has been successfully completed i times. Then $R^{(i)}_{n-1/i} = (D-S_{i-1/0})_{n-1/i}$, where $0 \leq i \leq n-2$.

Proof: Let i = 0. Then

$$R^{(0)}_{n-1/0} = D_{n-1/0} \qquad \text{; by definition, initially dest(j) = D (for S = j)}$$

$$= (D - 0)_{n-1/0}$$

$$= (D - S_{-1/0})_{n-1/0} \qquad \text{; because } S_{-1/0} = 0$$

The remainder of the proof is by induction on i.

Basis step: Let i = 1.

Case 1: $S_0 = D_0$. In this case the algorithm does not change $R^{(0)}$.

$$R^{(1)}_{n-1/1} = R^{(0)}_{n-1/1} \qquad \text{; because } S_0 = D_0 \text{ the test in line 7 of Algorithm 9.1 is false, so REQUEST is not executed}$$

$$= D_{n-1/1}$$

$$= (D - S_{0/0})_{n-1/1}$$

Case 2: $S_i \neq D_0$. In this case the algorithm for use with positive dom-

inant tags subtracts 2^0 from D.

Subcase (a): $S_0 = 0$, $R_0^{(0)} = D_0 = 1$.

$R_{n-1/1}^{(1)} = (R_{n-1/0}^{(0)} - 2^0)_{n-1/1}$; because $R_{n-1/0}^{(0)} = D_{n-1/0}$ and $S_0 \neq D_0$ the test in line 7 of Algorithm 9.1 is true, so REQUEST is executed

$= (R_{n-1/1}^{(0)} + 2^0 - 2^0)_{n-1/1}$; since $R_0^{(0)} = 1$

$= R_{n-1/1}^{(0)}$

$= D_{n-1/1}$

$= (D - S_{0/0})_{n-1/1}$; since $S_0 = 0$

Subcase (b): $S_0 = 1$, $R_0^{(0)} = D_0 = 0$.

$R_{n-1/1}^{(1)} = (R_{n-1/0}^{(0)} - 2^0)_{n-1/1}$

$= (D_{n-1/0} - S_0)_{n-1/1}$; since $S_0 = 1$

$= (D - S_{0/0})_{n-1/1}$

Induction step: Assume $R^{(i)}_{n-1/i} = (D - S_{i-1/0})_{n-1/i}$.

Case 1: $S_i = R^{(i)}_i$. In this case the algorithm does not change $R^{(i)}$.

$R^{(i+1)}_{n-1/i+1} = R^{(i)}_{n-1/i+1}$; because $S_i = R^{(i)}_i$ the test in line 7 of Algorithm 9.1 is false, so REQUEST is not executed

$= (D - S_{i-1/0})_{n-1/i+1}$

$= (D - S_{i/0})_{n-1/i+1}$; since $S_i = R^{(i)}_i = (D - S_{i-1/0})_i$ no borrow can propagate into the i+1 bit position

Case 2: $S_i \neq R^{(i)}_i$. In this case the algorithm for use with positive dominant tags subtracts 2^i from $R^{(i)}$.

Subcase (a): $S_i = 0$, $R^{(i)}_i = 1$.

$R^{(i+1)}_{n-1/i+1} = (R^{(i)}_{n-1/i} - 2^i)_{n-1/i+1}$; because $S_i \neq R^{(i)}_i$ the test in line 7 of Algorithm 9.1 is true, so REQUEST is executed

$= (R^{(i)}_{n-1/i+1} + 2^i - 2^i)_{n-1/i+1}$; since $R^{(i)}_i = 1$

$= R^{(i)}_{n-1/i+1}$

$= (D - S_{i-1/0})_{n-1/i+1}$

$$= (D - S_{i/0})_{n-1/i+1} \qquad \text{; since } S_i = 0$$

Subcase (b): $S_i = 1$, $R_i^{(i)} = 0$.

$$R_{n-1/i+1}^{(i+1)} = (R_{n-1/i}^{(i)} - 2^i)_{n-1/i+1}$$

$$= ((D - S_{i-1/0})_{n-1/i} - 2^i)_{n-1/i+1}$$

$$= ((D - S_{i/0})_{n-1/i})_{n-1/i+1} \qquad \text{; since } S_i = 1$$

$$= (D - S_{i/0})_{n-1/i+1} \cdot$$

[]

Let $|T|$ denote the magnitude of T. For positive dominant routing tags: $|T| = |D-S| = D-S$ where $D \geq S$, and when $D < S$, $|T|$ = the two's complement of $|D-S|$ = the two's complement of $(S-D) = N-(S-D) = N+(D-S)$. Thus, $T_i = (D-S_{i-1/0})_i - S_i$. Hence, $T_i = (D-S_{i-1/0})_i \oplus S_i$. From Theorem 9.1, $R_i^{(i)} = (D-S_{i-1/0})_i$. Therefore, $T_i = R_i^{(i)} \oplus S_i$. So T_i and $R_i^{(i)} \oplus S_i$ are equivalent criteria. When $R_i^{(i)} \oplus S_i = 0$, that is, when $R_i^{(i)}$ and S_i are the same, the algorithm selects the straight connection in stage i as does the routing tag. When $R_i(i)$ and S_i differ, the algorithm selects the $+2^i$ connection, as would the routing tag, since in this case $T_i = 1$. Thus the algorithm faithfully simulates the operation of the positive dominant permutation routing tags.

If negative dominant permutation routing tags are to be used, a similar argument shows that the algorithm, with the appropriate version of REQUEST, simulates this situation accurately. For natural permuta-

tion routing tags, REQUEST includes a check for positive dominance by determining if dest(j) - j is positive, i.e. greater than zero. If so, dest(j) is updated using the positive dominant PM2I connection. If not, dest(j) is revised using the negative dominant PM2I link. All other operations are as in the previous two cases.

Correct operation with natural routing tags is assured because the revised elements of the dest vector retain the positive dominant versus negative dominant information. This is so because if S is routed to D using the natural tag, and A_i is the stage i output cell that the specified path maps to in stage i, then $S \leq A_i \leq D$ when $D > S$, and $S \geq A_i \geq D$ when $D < S$, for $0 \leq i < n$. So the dominance of the natural routing tag generated by S and D can be determined from S and any of the A_i.

The time complexity of the algorithm is $O(N \log_2 N)$, in all cases. The space complexity is O(N), in particular, two N-element vectors.

9.3 Conclusions

Several routing tag schemes for the ADM in an SIMD environment were reviewed in this chapter. Full routing tags, which allow unrestricted use of the capabilities of the ADM network, were described and the difficulty of generating such tags was noted. Natural, positive dominant, and negative dominant permutation routing tags, which are easily computed, were defined.

An algorithm, in three versions, was given to determine the passability of an arbitrary overall permutation using any of these three types of more easily computed tags. Algorithm operation was described and the computed determination of permutation passability shown to be correct.

CHAPTER 10

FURTHER PROPERTIES OF THE AUGMENTED DATA MANIPULATOR NETWORK

10.1 Introduction

This chapter continues the development of ADM network properties. Group theory [MCC], an area of abstract algebra, is used to prove the theorems which are presented. The results obtained further characterize the permuting ability of the ADM network.

10.2 Definitions and Notation

The results to be presented make use of the following terminology and notation from group theory. For some permutation f, f^{-1} is the inverse of f, that is if f maps (connects) a source S to a destination D, then f^{-1} maps D to S. This holds for all source/destination pairs of f.

A permutation, f, on the set of PE addresses, {0,1,...,N-1}, can be represented in Cauchy notation as

$$\begin{pmatrix} 0 & 1 & \dots & N-1 \\ f(0) & f(1) & \dots & f(N-1) \end{pmatrix}$$

where the top line is the source and the bottom line is the destination to which f maps the source.

The permutation f can be represented as a product of cycles, where the cycle

AD-A096 399 PURDUE UNIV LAFAYETTE IN SCHOOL OF ELECTRICAL ENGINEERING F/G 9/2
PROPERTIES OF THE AUGMENTED DATA MANIPULATOR NETWORK IN A SIMD --ETC(U)
DEC 80 G B ADAMS, H J SIEGEL AFOSR-78-3581

UNCLASSIFIED TR-EE 80-51 AFOSR-TR-81-0203 NL

2 OF 2
AD A 096399

END
DATE FILMED
4-81
DTIC

$$(j_0\ j_1\ j_2 \cdots j_{x-1}\ j_x)$$

means $f(j_0) = j_1$, $f(j_1) = j_2$, ..., $f(j_{x-1}) = j_x$, and $f(j_x) = j_0$. The physical interpretation of this cycle is that network input j_0 is connected to network output j_1, input j_1 is connected to output j_2, ..., input j_{x-1} is connected to output j_x, and input j_x is connected to output j_0. The length of this cycle is x+1; it is called an x+1 cycle. A transposition is a cycle of length two. For example, if f is written in Cauchy notation as

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 0 & 4 & 5 & 7 & 6 \end{pmatrix}$$

then it can be written as the product of the four cycles:

$$(0\ 1\ 2\ 3)(4)(5)(6\ 7)\ .$$

A permutation is even or odd depending on whether it can be expressed as a product of an odd or even number of transpositions, respectively. For example, the permutation represented as (0 1 2) can be written as (1 2)(0 2), where the product is formed from right to left, as is the customary notation. Alternately, a k-cycle can be shown to be even or odd as k is odd or even.

An element of a cycle is any network address contained in the cycle. Two cycles are disjoint if they have no elements in common. The cycles (0 1 6) and (7 3) are disjoint, for example. Any permutation can be written as a unique product of disjoint cycles. This is the disjoint cycle decomposition of the permutation.

10.3 Theoretical Results

This section considers further properties of the ADM network.

Lemma 10.1: Any 1-cycle in an overall permutation must be routed using the straight connection on every stage.

Proof: Proof by contradiction. Assume a PM2I link is used in stage 0. This must give a mapping to a destination which differs from the source in bit 0 since only stage 0 can affect this bit. So the straight connection must be used in stage 0. Using a PM2I link in stage 1 must give a mapping to a destination differing from the source in bit 1, because only stage 1 can affect bit 1 once stage 0 is fixed. Continuing this chain of reasoning shows each stage must use the straight link.

[]

In [SI6] it is shown that in the i^{th} stage of the ADM network, $0 \leq i < n$, the transfer of data from the stage i input cell j can be represented by only one of five possible cycles. The cycles are (j), (j $j+2^i$ $j+2*2^i$ $j+3*2^i$ ... $j+N-2^i$), (j+N-2^i ... $j+3*2^i$ $j+2*2^i$ $j+2^i$ j), (j $j+2^i$), or (j $j-2^i$) where all arithmetic is modulo N, $0 \leq j < N$. The first of these is a 1-cycle. The network can perform any 1-cycle at any stage. The next two can be found to be 2^{n-i} cycles. These specific 2^{n-i} cycles are called network implemented 2^{n-i}-cycles since they are directly implemented by stage i. The last two cycles listed are transpositions. These are called network implemented transpositions. Collectively, the cycles which can be directly implemented by a stage are referred to as network implemented.

The relationship between ADM network structure and the cycle decomposition of a permutation is considered in the following theorem and corollary.

Theorem 10.1: In the ADM network all cycles of length greater than one which are part of the unique disjoint cycle decomposition of a passable overall permutation, must be expressible in terms of a product of network implemented transpositions and/or 2^{n-i}-cycles with elements limited to those of the cycles of length greater than one.

Proof: From Lemma 10.1, the network must be set to straight at each stage i cell with the same address as any 1-cycles in the overall permutation. Thus, the element of any 1-cycle cannot appear as an element in any n-cycle, for $n > 1$, of the passable overall permutation.

[]

Corollary 10.1: An overall permutation consisting of a single transposition is passable by the ADM network if and only if it is a network implemented transposition, i.e., of the form (j $j \pm 2^i$ modulo N), where $0 \leq j < N$ and $0 \leq i < n$.

Proof: From Theorem 10.1, only the network implemented cycles whose elements are the same as those of the given single transposition can be used to pass the permutation. That is, if more than two cells use PM2I-type links then N-2 routes of only straight links cannot exist. Thus, if an overall permutation consisting of a single transposition is passable then that transposition is network implemented.

If an overall permutation consists of a single transposition which is network implemented then, clearly, it is passable.

[]

S_N is the permutation group for N elements. That is S_N is the set of all permutations of N elements. A_N is the alternating group on N objects and consists of all even permutations of the N items.

Theorem 10.2: The ADM network cannot perform all 3-cycles in one pass for $N \geq 8$.

Proof: Consider the 3-cycle (0 1 6). To be passable it must decompose into network implemented cycles with elements in the set {0,1,6} (see Theorem 10.1). The network implemented cycles must be of length three or shorter since there are only three allowable elements. So only network implemented transpositions may be used.

The possible transpositions with elements in the set {0,1,6} are (0 1), (0 6), and (1 6). Network implemented transpositions are of the form (j $j \pm 2^i$ modulo N), so (1 6) is not implemented for any N. The cycle (0 6) is implemented only when N = 8. When N = 8, $0-2^1$ modulo 8 = 6, so (0 6) is implemented. For N < 8, (0 6) has an element, 6, outside the range of source addresses, 0 to N-1. For N > 8 (i.e., n > 3), $0+2^i$ modulo $N \neq 6$, for $0 \leq i < n$, and

$$0\text{-}2^i \text{ modulo } N = 2^n - 2^i$$
$$= \sum_{k=0}^{n-1} 2^k - \sum_{k=0}^{i-1} 2^k$$
$$= \sum_{k=i}^{n-1} 2^k$$
$$\neq 6$$

Thus for $N > 8$, (0 1 6) clearly cannot be performed since only (0 1) can be performed of the three possible transpositions on the set {0,1,6}. For $N = 8$, the transpositions (0 1) and (0 6) can be performed. However, (0 6) is performed in stage 1 and (0 1) in stage 0 only. So the network can only implement the product (0 1)(0 6) and not (0 6)(0 1) since stage order is fixed. But (0 1)(0 6) = (0 6 1) $\neq$ (0 1 6). The ADM cannot, then, pass all 3-cycles.

[]

Theorem 10.3: For $N \geq 8$ the ADM network does not pass A_N.

Proof: From Theorem 10.2 the ADM does not pass all 3-cycles for $N \geq 8$. A_N contains all even permutations of N elements and a 3-cycle is an even permutation. Thus the ADM does not pass every element of A_N.

[]

10.4 Conclusions

This chapter presented further properties of the ADM network in an SIMD environment. The results which were stated aid in characterizing the family of permutations passable by the ADM.

CHAPTER 11

CONCLUSIONS

This work is a study of various aspects of the ADM network which influence its suitability for use in SIMD parallel processing systems. The first aspect considered was the number of permutations passable by the ADM network. Next a routing tag scheme that has been developed for distributed control of the ADM network was described so that its performance in an SIMD environment can be investigated. Then algorithms for determining permutation passability using these routing tags were presented and analyzed. Finally, some additional theoretical results were developed.

SIMD machine models were described in Chapter 2 to provide a background in which to evaluate the ADM network. The role of the interconnection network in SIMD machines was outlined for two system architectures. Some basic requirements and limitations for each structure were noted.

Chapter 3 introduced PASM, a partitionable SIMD/MIMD multimicroprocessor system. Unique features of PASM include being 1) dynamically reconfigurable; 2) able to operate in either SIMD or MIMD mode of parallelism; and 3) able to be partitioned into machines of different sizes, each of which may operate in SIMD or MIMD mode. Two interconnection networks are being considered for use in PASM: the Generalized Cube and the ADM. The Generalized Cube network is better understood

than the ADM network. Increased knowledge of ADM properties will allow a more informed decision on the interconnection network for PASM, and other parallel processing systems, to be made.

The Generalized Cube and ADM networks were formally defined in Chapter 4. The Generalized Cube was noted to be representative of a class of cube-type networks, including the STARAN flip, the omega, the indirect binary n-cube, and the SW-banyan (S=F=2) networks. The ADM was shown to be derived from the data manipulator.

The number of distinct permutations passable by the Generalized Cube was given in Chapter 5. The procedure used to obtain this result relies on the one-to-one correspondence between permutations and legitimate network settings. Both the count of permutations and this one-to-one correspondence were used later for comparative purposes.

Chapter 6 considered the number of permutations performable by the ADM network. A method was given for counting the number of settings which are permutations for any stage. Using partitioning theory and combinatorial mathematics, upper and lower bounds were established on the number of overall permutations which an ADM network can perform. To assess the characteristics of the bounds their tightness and asymptotic behavior was studied. For an ADM network with eight inputs, an exact count of the number of passable overall permutations was proven. Lastly, the number of ADM passable permutations was compared with that of the Generalized Cube.

The use of routing tags for distributed control of interconnection networks was introduced in Chapter 7. The permuting ability of the Generalized Cube when used with routing tags was discussed. The results

obtained were used for comparison with the ADM network.

In Chapter 8 several routing tag schemes which allow distributed control of the ADM network were reviewed. The number of permutations performable using either positive dominant or negative dominant permutation routing tags was counted.

Chapter 9 presented an algorithm which can determine if an arbitrary overall permutation is passable using either natural, positive dominant, or negative dominant permutation routing tags. Correct operation of the algorithm was demonstrated and its complexity stated.

Further properties of the ADM network can be derived using group theory. The results which were presented in Chapter 10 aid in characterizing the family of permutations passable by the ADM network.

Choosing the interconnection network for a parallel processing system such as PASM is an important and difficult design task for the system architect. A satisfactory compromise among many interconnection network parameters including, among others, permuting capability and performance with distributed control, must be reached. Analyses such as those presented here are necessary in order to evaluate the cost-effectiveness of the ADM as an SIMD interconnection network.

LIST OF REFERENCES

[BA] K. E. Batcher, "The flip network in STARAN," 1976 International Conference on Parallel Processing, Aug. 1976, pp. 65-71.

[BOU] W. J. Bouknight, S. A. Denenberg, D. E. McIntyre, J. M. Randall, A. H. Sameh, and D. L. Slotnick, "The Illiac IV system," Proceedings of the IEEE, Vol. 60, pp. 369-388, Apr. 1972.

[FE] T. Feng, "Data manipulating functions in parallel processors and their implementations," IEEE Transactions on Computers, Vol. C-23, Mar. 1974, pp. 309-318.

[FL] M. J. Flynn, "Very high-speed computing systems," Proceedings of the IEEE, Vol. 54, Dec. 1966, pp. 1901-1909.

[FSS] A. E. Feather, L. J. Siegel, and H. J. Siegel, "Image correlation using parallel processing," Fifth International Conference on Pattern Recognition, Dec. 1980, pp. 503-507.

[GL] G. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," First Annual Symposium on Computer Architecture, Dec. 1973, pp. 21-28.

[JE] C. Jensen, "Taking another approach to supercomputing," Datamation, Vol. 24, Feb. 1978, pp. 159-172.

[LA] D. Lawrie, "Access and alignment of data in an array processor," IEEE Transactions on Computers, Vol. C-24, Dec. 1975, pp. 1145-1155.

[LE] J. Lenfant, "Parallel permutations of data: a Benes network control algorithm for frequently used permutations," IEEE Transactions on Computers, Vol. C-27, July 1978, pp. 637-647.

[MAS] R. J. McMillen, G. B. Adams III, and H. J. Siegel, "Permuting with the augmented data manipulator network," Eighteenth Annual Allerton Conference on Communication, Control, and Computing, Oct. 1980, to appear in the proceedings.

[MCC] N. H. McCoy, Introduction to Modern Algebra (3rd edition), Boston: Allyn and Bacon, 1975.

[MCM] R. J. McMillen and H. J. Siegel, Interconnection Networks and Operating System Considerations for PASM - A Reconfigurable Multimicroprocessor System, School of Electrical Engineering, Purdue University, Technical Report TR-EE 80-15, June 1980, 177 pp.

[MS] R. J. McMillen and H. J. Siegel, "MIMD machine communications using the augmented data manipulator network," Seventh Annual Symposium on Computer Architecture, May 1980, pp. 51-58.

[MSS1] P. T. Mueller, Jr., L. J. Siegel, and H. J. Siegel, "A parallel language for image and speech processing," Fourth International Computer Software and Applications Conference (COMPSAC '80), Oct. 1980, pp. 476-483.

[MSS2] P. T. Mueller, Jr., L. J. Siegel, and H. J. Siegel, "Parallel algorithms for the two-dimensional FFT," Fifth International Conference on Pattern Recognition, Dec. 1980, pp. 497-502.

[OS] M. J. O'Donnell and C. H. Smith, "A combinatorial problem concerning processor interconnection networks," Department of Computer Sciences, Purdue University, Technical Report CSD TR-352, Nov. 1980, 5 pp.

[PE] M. C. Pease, "The indirect binary n-cube microprocessor array," IEEE Transactions on Computers, Vol. C-26, May 1977, pp. 458-473.

[SI1] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effect of processor address masks," IEEE Transactions on Computers, Vol. C-26, Feb. 1977, pp. 153-161.

[SI2] H. J. Siegel, "Controlling the active/inactive status of SIMD machine processors," 1977 International Conference on Parallel Processing, Aug. 1977, p. 183.

[SI3] H. J. Siegel, "Preliminary design of a versatile parallel image processing system," Third Biennial Conference on Computing in Indiana, Apr. 1978, pp. 11-25.

[SI4] H. J. Siegel, "Interconnection networks for SIMD machines," Computer, Vol. 12, June 1979, pp. 57-65.

[SI5] H. J. Siegel, "A model of SIMD machines and a comparison of various interconnection networks," IEEE Transactions on Computers, Vol. C-28, Dec. 1979, pp. 907-917.

[SI6] H. J. Siegel, "The theory underlying the partitioning of permutation networks," IEEE Transactions on Computers, Vol. C-29, Sep. 1980, pp. 791-801.

[SI7] L. J. Siegel, "Image processing on a partitionable SIMD machine," Workshop on New Computer Architectures and Image Processing, June 1980, to appear.

[SKW] H. J. Siegel, F. Kemmerer, and M. Washburn, "Parallel memory system for a partitionable SIMD/MIMD machine," 1979 International Conference on Parallel Processing, Aug. 1979, pp. 212-221.

[SM1] H. J. Siegel and P. T. Mueller, Jr., "The organization and language design of microprocessors for an SIMD/MIMD system," Second Rocky Mountain Symposium on Microcomputers, Aug. 1978, pp. 311-340.

[SM2] H. J. Siegel and R. J. McMillen, "The use of the augmented data manipulator in PASM," Fourteenth Annual Hawaii International Conference on System Sciences, to appear, Jan. 1981.

[SMS1] H. J. Siegel, P. T. Mueller, Jr., and H. E. Smalley, Jr., "Control of a partitionable multimicroprocessor system," 1978 International Conference on Parallel Processing, Aug. 1978, pp. 9-17.

[SMS2] L. J. Siegel, P. T. Mueller, Jr., and H. J. Siegel, "FFT algorithms for SIMD machines," Seventeenth Allerton Conference on Communications, Control, and Computing, Oct. 1979, pp. 1006-1015.

[SS1] H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks," Fifth Annual Symposium on Computer Architecture, Apr. 1978, pp. 223-229.

[SS2] S. D. Smith and H. J. Siegel, "Recirculating, pipelined, and multistage SIMD interconnection networks," 1978 International Conference on Parallel Processing, Aug. 1978, pp. 206-214.

[SS3] S. D. Smith and H. J. Siegel, Design and Analysis of Interconnection Networks for Partitionable Parallel Processing Systems, School of Electrical Engineering, Purdue University, Technical Report TR-EE 79-39, Aug. 1979, 274 pp.

[SSE] P. H. Swain, H. J. Siegel, and J. El-Achkar, "Multiprocessor implementation of pattern recognition: a general approach," Fifth International Conference on Pattern Recognition, Dec. 1980, pp. 309-317.

[SSKMS] H. J. Siegel, L. J. Siegel, F. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, Jr., and S. D. Smith, PASM: A Partitionable Multimicrocomputer SIMD/MIMD System for Image Processing and Pattern Recognition, School of Electrical Engineering, Purdue University, Technical Report TR-EE 79-40, Aug. 1979, 69 pp.

[SSMA] S. D. Smith, H. J. Siegel, R. J. McMillen, and G. B. Adams III, "Use of the augmented data manipulator multistage network for SIMD machines," 1980 International Conference on Parallel Processing, Aug. 1980, pp. 75-78.

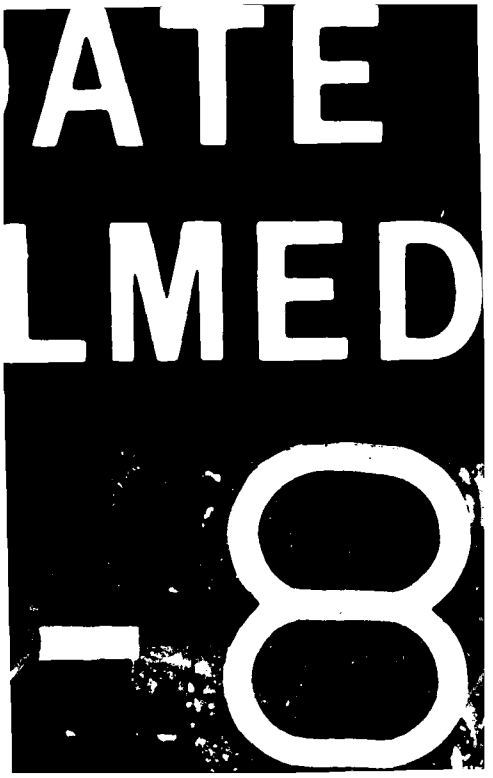
[SSMMS] H. J. Siegel, L. J. Siegel, R. J. McMillen, P. T. Mueller, Jr., and S. D. Smith, "An SIMD/MIMD multimicroprocessor system for image processing and pattern recognition," 1979 IEEE Computer Society Conference on Pattern Recognition and Image Processing, Aug. 1979, pp. 214-224.

[ST] H. S. Stone, "Parallel Computers," in Introduction to Computer Architecture, Science Research Associates, Inc., Chicago, 1975, pp. 318-374.

[WE] K. Y. Wen, Interprocessor Connections - Capabilities, Exploitation, and Effectiveness, Doctoral Thesis, Department of Computer Science, University of Illinois, Report UIUCDCS-R-76-830, Oct. 1976.

[WF1] C. Wu and T. Feng, "Fault diagnosis for a class of multistage interconnection networks," 1979 International Conference on Parallel Processing, Aug. 1979, pp. 269-278.

[WF2] C. Wu and T. Feng, "On a class of multistage interconnection networks," IEEE Transactions on Computers, Vol. C-29, Aug. 1980, pp. 694-702.


ATE
LMED
-8